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

NOTICE

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.

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

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

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.

DESIGN FOR TESTABILITY GUIDELINES

(Intentionally left blank)

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,

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

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

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

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

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

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

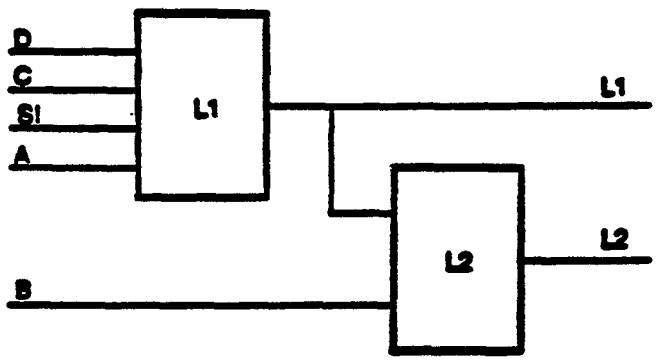
SUMMARY

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.

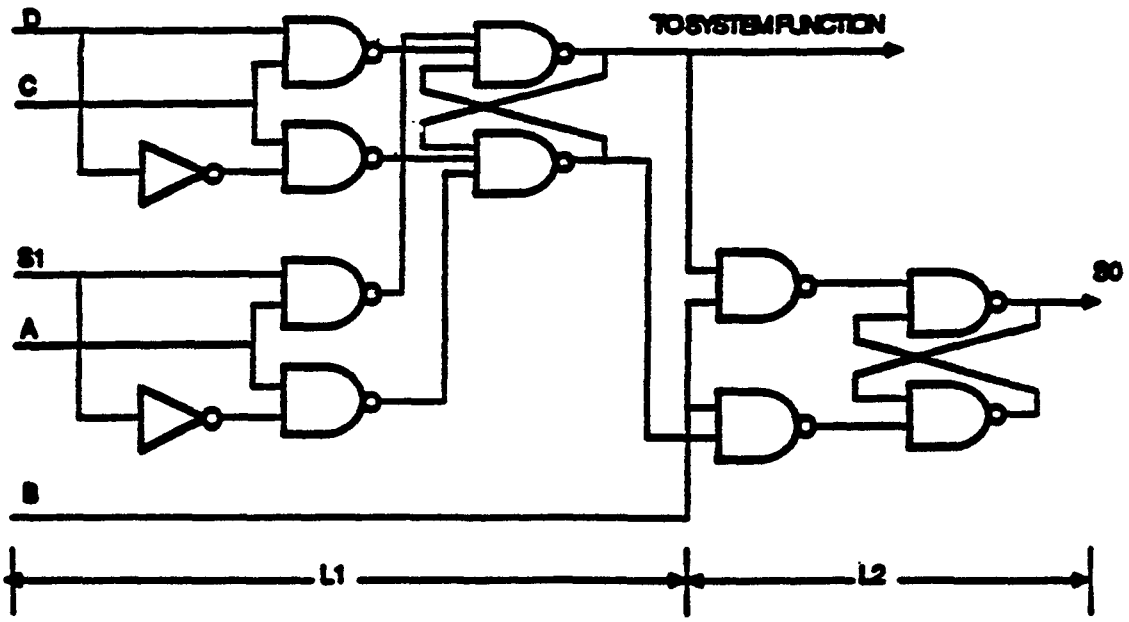
DESIGN FOR TESTABILITY GUIDELINES

Figure 3-1
Shift Register Latch (SRL)

3-1a. Symbolic representation

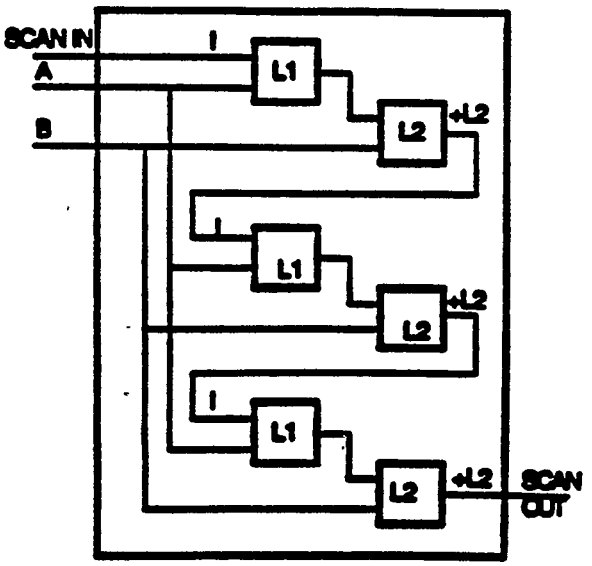


3-1b. Gate level implementation



DESIGN FOR TESTABILITY GUIDELINES

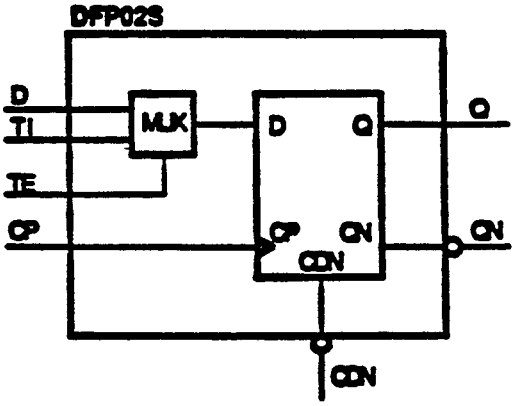
3-1c. Circuit interconnection



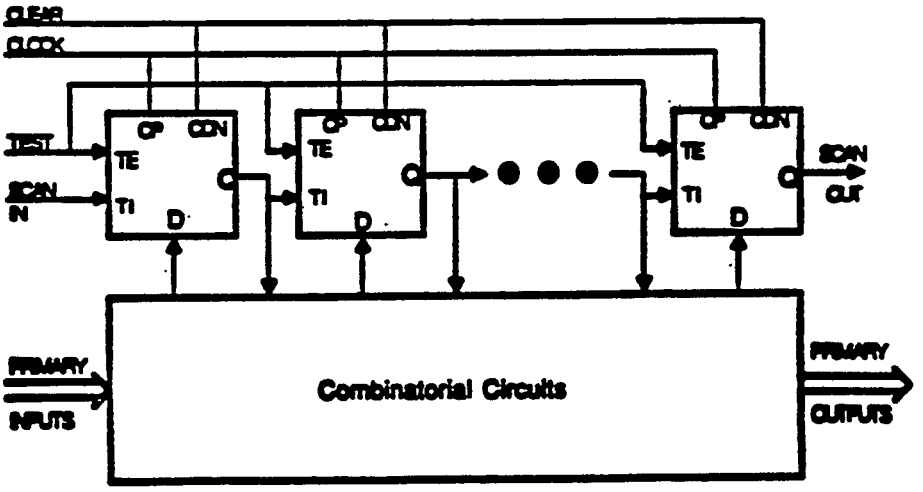
DESIGN FOR TESTABILITY GUIDELINES

Figure 3-2
Scan Path Implementation

3-2a. D Flip-flop with scan



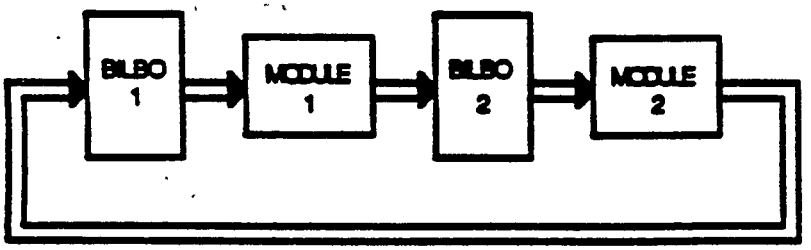
3-2b. Circuit implementation



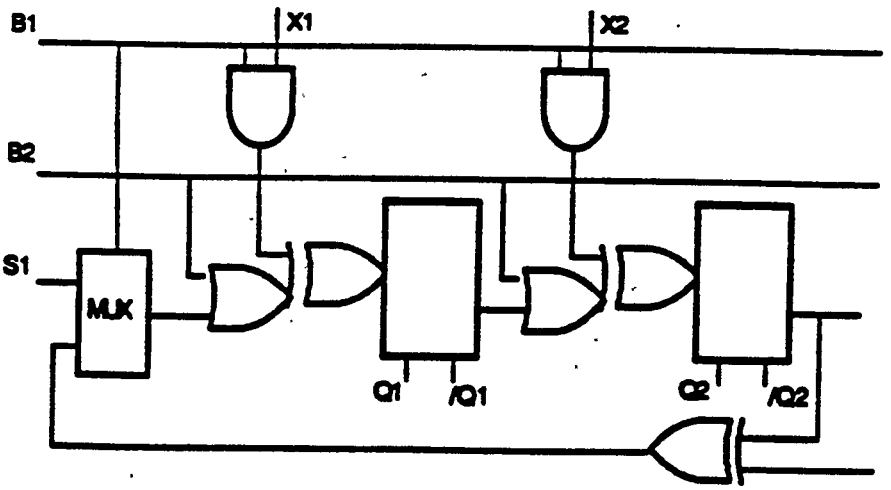
DESIGN FOR TESTABILITY GUIDELINES

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 capability

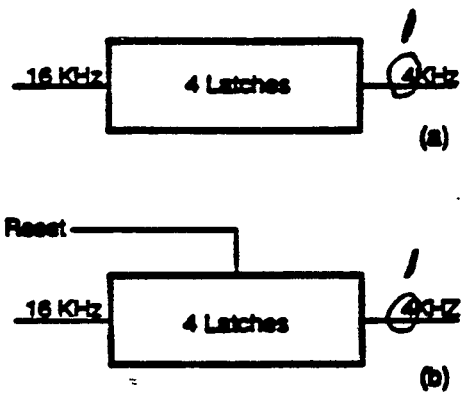
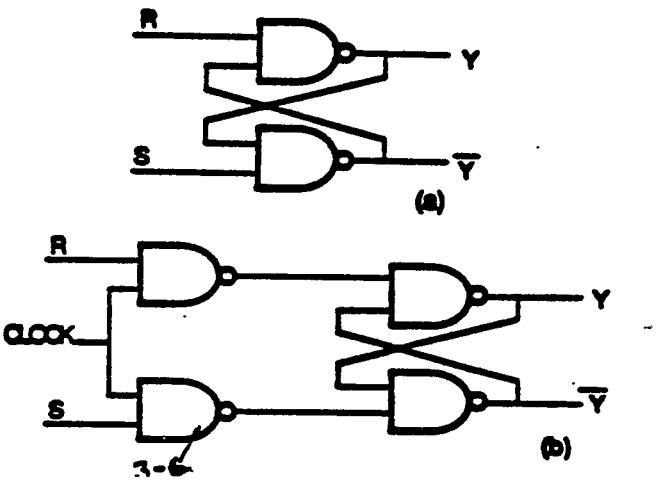
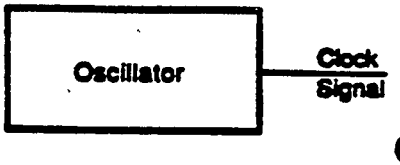


Figure 3-5
Asynchronous SR Latch

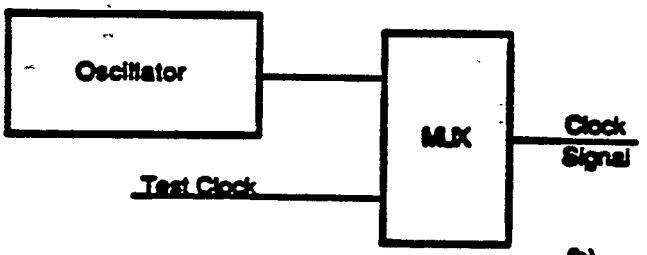


DESIGN FOR TESTABILITY GUIDELINES

Figure 3-6
Bypass internal clock



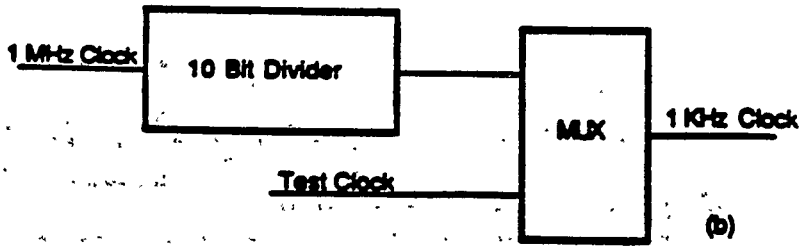
(a)



(b)

Figure 3-7
Bypass internal registers

(a)



(b)

DESIGN FOR TESTABILITY GUIDELINES

Figure 3-8
Partition long chain of sequential circuits

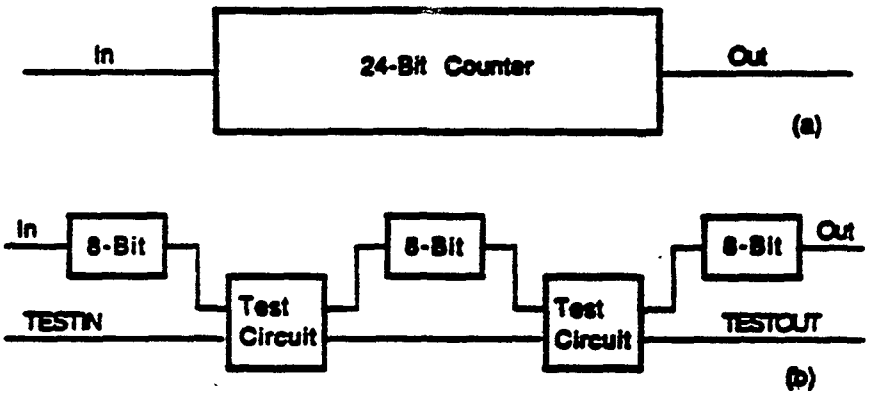
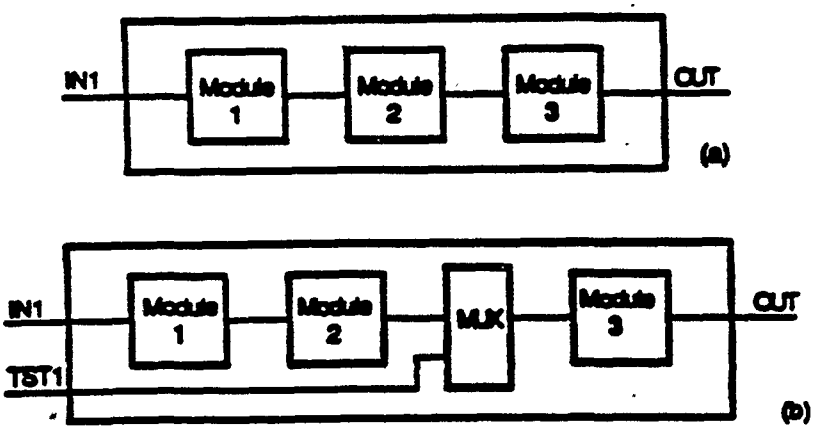


Figure 3-9
Direct access of functional module



DESIGN FOR TESTABILITY GUIDELINES

Figure 3-10
Observing functional modules

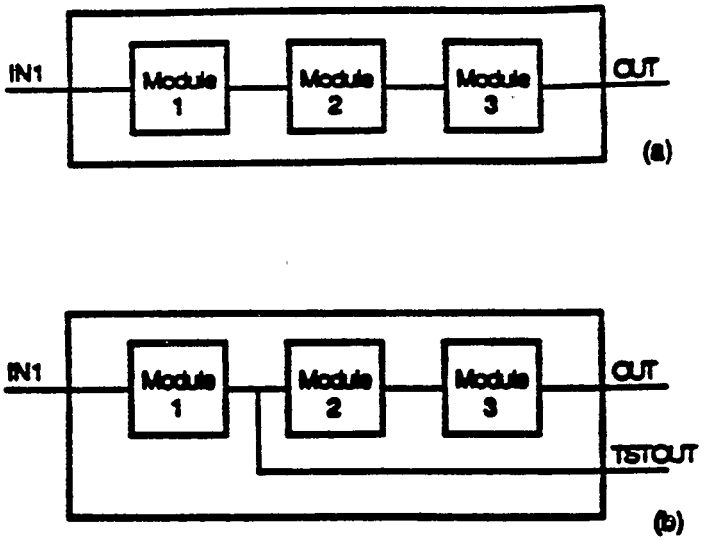
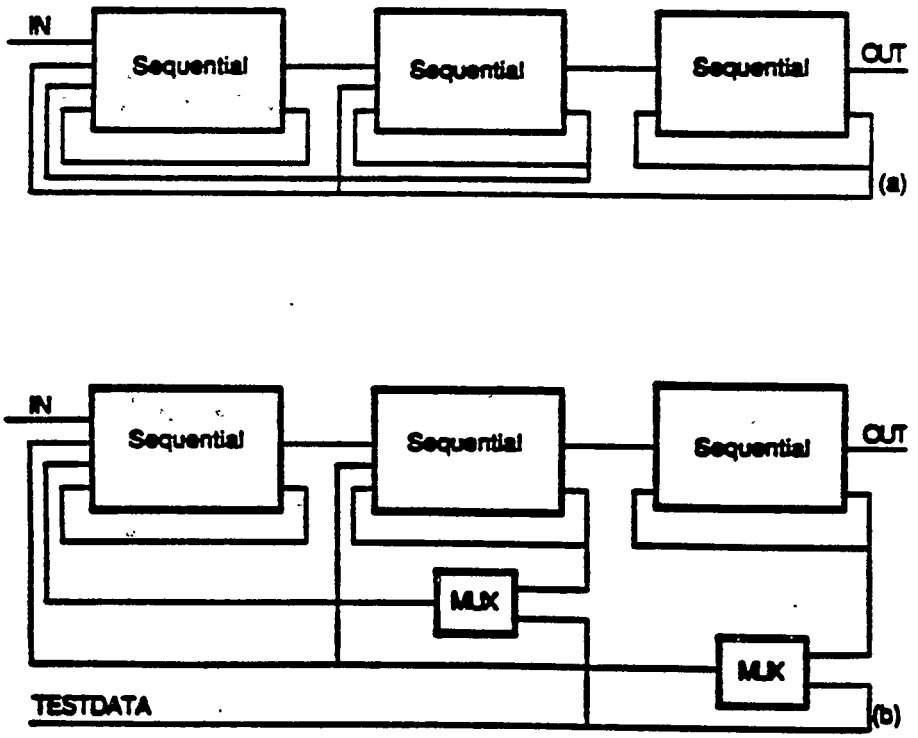


Figure 3-11
Nesting of sequential circuits



DESIGN FOR TESTABILITY GUIDELINES

Figure 3-12
Redundant circuitry

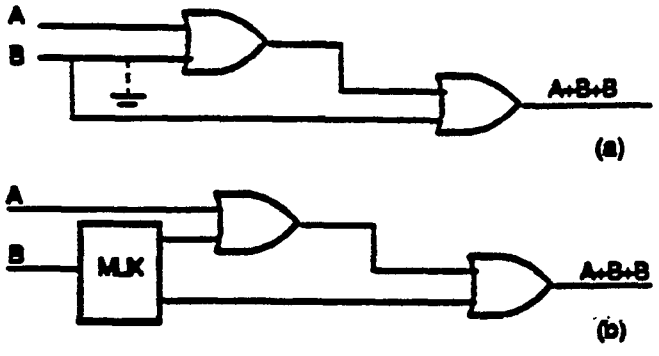
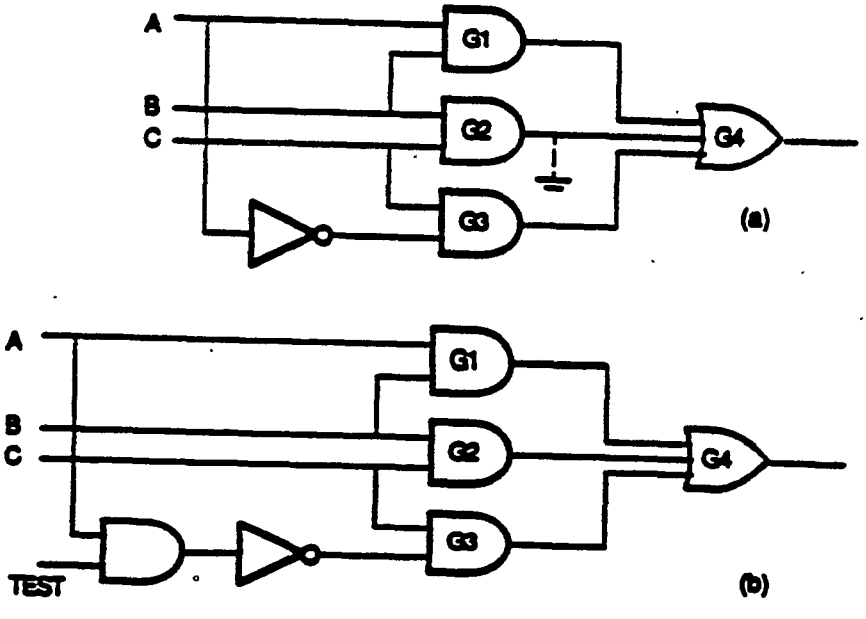


Figure 3-13
Reconverging Signals



DESIGN FOR TESTABILITY GUIDELINES

CHAPTER 4

ISOLATION TECHNIQUES FOR MEGACELL/MEMORY/ANALOG STRUCTURES IN SEMICUSTOM IC'S TO ENHANCE TESTABILITY

This chapter addresses a technique for isolating large circuit blocks in Semicustom IC's, in order to reduce the complexity and cost associated with performing functional testing of the chip. For the purpose of illustration, this discussion assumes that the chip to be tested contains the following large blocks:

* RAM
* ROM
* Random logic circuit
* Microprocessor
* Analog circuit (A/D converters)

In general, when these blocks are used together on a chip, a bus architecture would be employed. If it were not, a more complex testing circuit would be needed.

Figure 4-1 is a block diagram containing the blocks listed above. Clocking circuitry has been omitted in the interest of simplicity. The inputs to the Pipeline Register/Scan Register contain both Scan-in and Test Control signals. These are the only 2 additional testing signals required. The outputs of this register consist of control and address signals connected to the Control and Address Bus, data signals connected to the Data Bus, and a control signal connected to the multiplexer for the A/D converters. The Primary Input/Primary Output (PI/PO) and Primary Input (PI) lines are directly accessible from outside the circuit. The Pipeline Register/Scan Register serves the dual purpose of a conventional pipeline register during normal operation and a scan-in register during testing. This register may, in fact, be very wide (40 bits, for example). The multiplexer for the A/D converters may consist of transmission gates. The other two blocks are the Signature Analyzer (SA) and Decoders.

DESIGN FOR TESTABILITY GUIDELINES

Once the architecture described above has been implemented, the following steps can be used for testing. It should be noted that the steps are independent and sequencing (after Step 1) can be arranged for convenience.

1. Set the Test Control Signal to Test Mode.

2. Microprocessor test: Scan in a word that instructs the microprocessor to perform a specific operation and send the result to the signature analyzer. Additional words can be scanned in to perform various operations. Knowledge of the control words causes predictable responses from the microprocessor. The Decoder circuitry should contain "good" signatures to be compared with the output of the Signature Analyzer. This will determine if the microprocessor is working correctly. A linear feed-back shift register could be used to design the Signature Analyzer.

3. ROM test: Scan words in that enable the ROM and allow the contents to be read out one byte at a time. The contents of the ROM can be monitored and compressed in the Signature Analyzer. Results of the analysis would determine the functionality.

4. RAM test: Scan words in that enable the RAM. Conventional RAM testing methods can be employed because address and data lines are directly connected to the RAM and other circuit blocks are disabled.

5. Random logic test: Scan in a random logic test word. Use conventional random logic testing methods. Scan registers may provide assistance in this area.

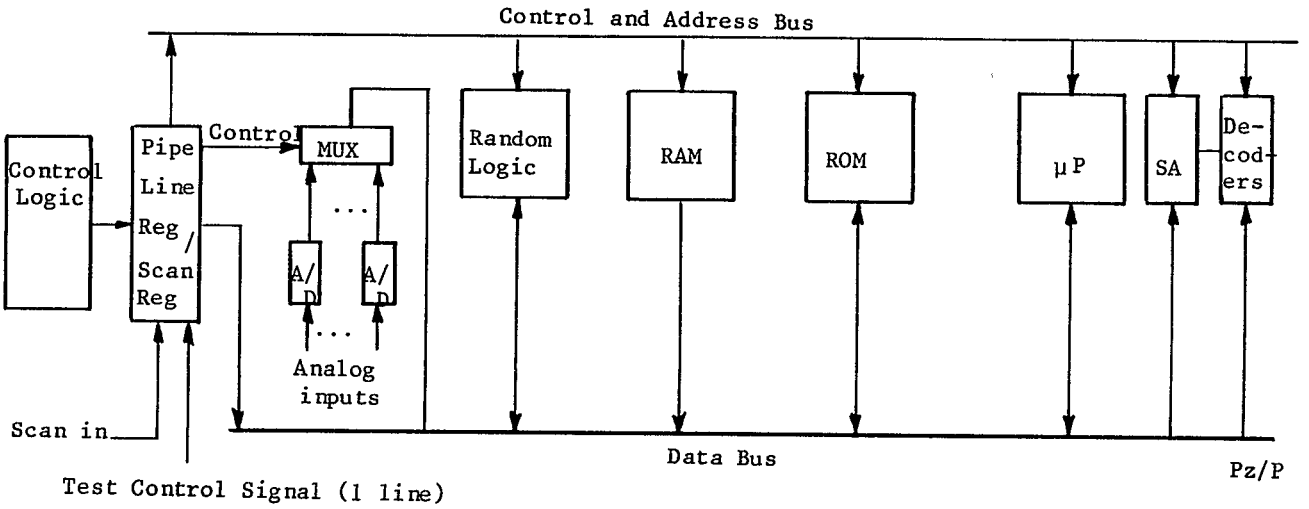
6. A/D converter test: Scan in a word that enables the multiplexer. Apply analog inputs to the A/D converters, one input at a time. The digitized analog inputs can be read out via the data bus and compared with predicted values.

With reference to performance degradation, data shows that a typical transmission gate contributes about 1.3k ohms and 0.5 picofarad. In one typical application the output slew rate may degrade to 67% and the propagation delay may increase by 33%.

The hardware overhead of this technique is reflected in the Signature Analyzer, Decoder, and the modifications to make the pipeline register serve a dual purpose. Only one Signature Analyzer is needed, since it can serve each of the other blocks. However, a separate Decoder will be needed for each block.

DESIGN FOR TESTABILITY GUIDELINES

FIGURE 4-1
BLOCK DIAGRAM OF ISOLATED LARGE CIRCUIT BLOCKS



DESIGN FOR TESTABILITY GUIDELINES

(Intentionally left blank)

DESIGN FOR TESTABILITY GUIDELINES

CHAPTER 5

AUTOMATIC TEST PATTERN GENERATION

This chapter discusses an increasingly popular test methodology called Automatic Test Generation (ATG). It is the intent of this discussion to provide a practical overview of today's circuit design environment and the way ATG integrated into this environment. Three ATG approaches for sequential circuits (full scan, partial scan, and non-scan) will be discussed. Important testing and debugging implications of using the ATG approach will also be presented.

The circuit design environment has evolved dramatically over the past few years, providing computing power and software to aid in almost every phase of the design process. Schematic capture, simulation, test extraction, and layout represent the essential backbone software programs for a complete design system. Semiconductor process technology has kept pace, permitting implementation of extremely complex logic systems on a single chip. This has resulted in an increase in the difficulty of chip testing.

Much of the added testing difficulty is due to higher density and fixed I/O count. Packing more logic onto one circuit greatly increases the gate to I/O ratio and decreases the accessability to that logic. Allowing a reasonable number of I/O signals to be used for testing can considerably ease the testing process. These test signals can be used to add controllability and/or observability to the circuit in a variety of ways. However, for a great many designs it is not possible to set aside the desired number of dedicated test pins due to pinout constraints. In any case, the task of generating a complete and efficient test program must be accomplished.

Most often, designers specify a functional simulation, and the test program is automatically extracted from the functional simulation results. Fault grading tools allow the designer to grade his test program by calculating a fault coverage percentage. The designer may then add to the test program to increase the fault coverage to an acceptable level. The automation of this "test generation" process is the subject of this chapter.

An integrated design environment begins with network entry (as shown in Figure 5-1) and proceeds with electrical rules checking and simulation. During the design process it is often necessary

to modify the network to correct errors and enhance the circuit. Once the design has been solidified, the process of test generation must be accomplished. It is important that testing considerations be addressed early in the design phase so test generation becomes a task of implementing the predefined testing plan. The designer should decide upon a target percentage for fault coverage as soon as possible, as a very high percentage may require the adoption of certain circuit design methods. It is possible that during the ATG phase, additional test points will be required to achieve the target percentage. If this is the case, the network will have to be modified and resimulated. For this reason, it is important in terms of efficiency and effectiveness that the ATG software be integrated into the design environment. ATG tools require the circuit description at a gate or transistor level, all circuit timing information, any pre-generated functional test patterns and any user specified parameters. The software will then use this information to follow a flow similar to that shown in Figure 5-2.

In this example, the first step is to check the circuit for adherence to any scan design rules imposed by the ATG tool. Fault simulation is then performed with the optional user generated functional patterns. All faults detected by these patterns are tagged and eliminated from the ATG process. Any remaining undetected faults are passed to the ATG algorithm where one fault is selected. Fault simulation is rerun to check the effects of generating the input patterns necessary to check the fault under consideration, and identify any additional undetected faults that may have been detected by this pattern. Iteration of this process continues until the desired fault coverage is achieved. Fault simulation, as an intermediate step, checks the targeted fault as well as all other undetected faults, thus reducing the number of ATG iterations.

DESIGN FOR TESTABILITY GUIDELINES

PATH SENSITIZATION

Test pattern generation programs use a variety of methods to achieve essentially the same test generation capability. The definition of a good test for a fault is an input combination which produces an incorrect output when the fault under consideration is present in the circuit. Thus, the presence of the fault can be observed at the outputs. Path sensitization is one such method of generating a test for a fault. The general procedure is:

1.) Select a fault (such as node "stuck at one").
2.) Assign the faulty wire a value opposite to the fault condition (such as node at "zero").
3.) Choose a path from the fault to a circuit output.
4.) Sensitize this path by assigning logic values to gate inputs along the path such that the faulty signal is passed to the circuit output.
5.) Obtain the test to detect the fault by determining the network inputs which will produce the desired values on gate inputs along the sensitized path.

Some variations on this path sensitization method are the Boolean Differences Method, the D-Algorithm, and the Truth Table Comparison Method. All of the above are effective methods for generating tests for faults in most combinational circuitry (without reconvergent fanout). However, they are not effective tools for ATG in sequential circuits. This has led to the development of a set of design rules to simplify test generation in sequential circuits. Level-sensitive scan design (LSSD) is one such design methodology that imposes design restrictions that allow the circuit to be forced into a test mode in which the ATG methods above can be used to generate tests for faults.

Another approach is that of random vector generation. This method has straightforward applications for some combinational circuitry but has not been effectively extended to sequential circuitry. Certain combinational circuits lend themselves nicely to this approach while others do not. By evaluating some simple circuit parameters such as the number of primary inputs, the number of levels of logic, and the average fan-in, a measure of circuit resistance to random ATG can be made. The number of random vectors required to reach the desired coverage can then be estimated based on this resistance factor.

DESIGN FOR TESTABILITY GUIDELINES

FULL SCAN DESIGNS

By far, the most popular ATG methodology follows a full scan approach. This calls for the circuit to follow rigid design rules that allow the circuit to be placed in a test mode such that all logic storage elements are connected to shift registers, breaking up the combinational circuitry. When this method is used, the circuit is placed in the test mode and a test pattern is shifted in, thereby defining all internal storage states. The circuit is then taken out of the test mod packages

DESIGN FOR TESTABILITY GUIDELINES

TESTING AND DEBUGGING CONSIDERATIONS

There are several types of problems that can cause a semiconductor circuit to fail. These include logic design and layout errors as well as mask and wafer fabrication defects. When performing test pattern generation, the design engineer must first determine the strategy of testing for all possible problems.

It is very important to remember that ATG patterns should only be used to test for non-consistent faults relating to the manufacturing process. Although ATG patterns may be an efficient tool for indicating the presence of a defect, they generally provide no information relative to the nature and location of the defect. Consequently, ATG patterns are not useful to the designer who is involved in the debugging process. Most ATG packages provide a cross reference listing that shows which faults are detected in each vector. This information may be of little use, however, because many faults may be detected in each vector and isolating the specific fault may be impossible.

It is also important to note that, in general, using a scan type design and an ATG test methodology will yield a much larger (longer) test program than will a functional test methodology. For example, a 6000 gate design, assuming 20% sequential circuitry, will have approximately 300 latches. Assuming that 532 test vectors per latch will achieve greater than 97% fault coverage, the tester program will require almost 160,000 tester cycles. In contrast, it is usually possible to achieve greater than 80% fault coverage when operating the circuit in a functional mode with fewer than 10,000 tester cycles. In order to accommodate the long test programs resulting from ATG, the tester environment should provide for scan test capability such that once a pattern is presented, the tester can automatically shift it into the scan chain.

A straightforward method of avoiding debugging and pattern-length problems is to begin the test program with a thorough functional test. ATG patterns can then be added to increase fault coverage to an acceptable level. Most ATG systems accept manually generated vector input. In this way, a fault grade of the functional patterns can be done before proceeding with automatic test generation. Then the remaining undetected faults can be worked on by the ATG system.

DESIGN FOR TESTABILITY GUIDELINES

CONCLUSION

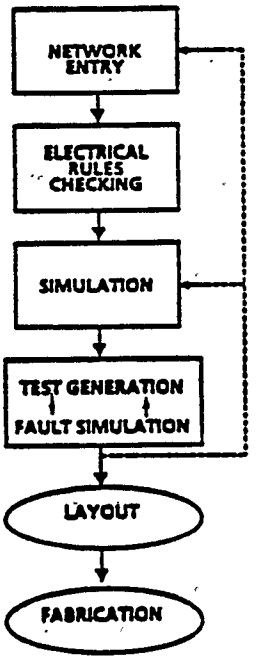
When embarking on an ATG methodology, the designer must consider many aspects of the total design environment:

* Integration of the ATG system into the design environment
* Adherence to circuit design rules pertaining to scan capability in the tester environment, if necessary
* Maintaining a functional test generation plan for debugging and critical path testing

Considering all of these items before adopting any particular ATG implementation should minimize any problems that might occur later.

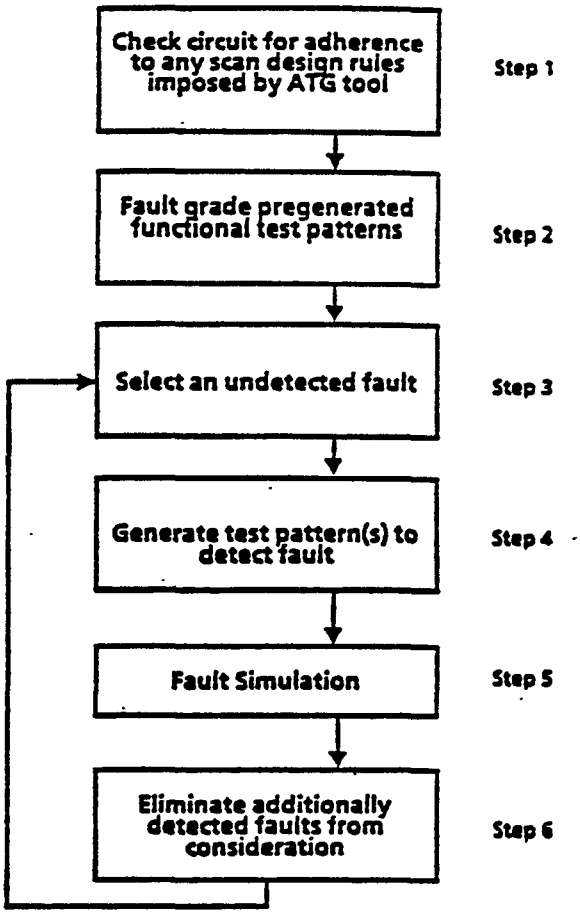
DESIGN FOR TESTABILITY GUIDELINES

Figure 5-1
Typical Design Flow



DESIGN FOR TESTABILITY GUIDELINES

Figure 5-2
Typical ATG Flow



DESIGN FOR TESTABILITY GUIDELINES

CHAPTER 6

TEST PROGRAM GENERATION FROM SIMULATION FILES

A logic truth table is used to electrically verify the functionality of an integrated circuit and thereby check for the presence of manufacturing defects. This truth table is referred to as a test pattern, and a test vector is defined as a line in the truth table where one or more inputs or outputs changes state. Each vector is stored in a single word of the tester's main memory, and the pattern to be applied to the DUT is loaded into the tester's local memory. If the number of vectors exceeds the size of the local memory, two or more memory loads controlled by the CPU would be required, thus increasing the test time. Minimizing the number of test vectors is desirable not only to reduce the test run time, but also to reduce simulation and fault grading times.

Logic simulators allow system designers to control and observe the behavior of all nodes in a circuit, if so desired. Furthermore, the designer can monitor propagation delays through internal circuitry in response to a change in input stimuli.

Neither one of these features is available in Automated Test Equipment (ATE) used to screen integrated circuits. Access to internal nodes is not possible, and propagation delays need to be measured from inputs to outputs of the device. Testing a semicustom chip is performed by applying binary patterns (input vectors) to stimulate the logic inputs and then comparing the actual output states to the response that was predicted during the simulation process.

In general, the phase relationship and sequence of input stimuli applied to an integrated circuit to test the functionality and performance of the device are somewhat different than the simulation patterns used to verify the logic design. These differences are, for the most part, due to the inherent characteristics of testing equipment used to evaluate logic devices.

Testers are synchronous systems that require all input vectors to be changed at fixed points within the test cycle. Figure 6-1 is a diagram of the phase relationship of input and output data with respect to the test cycle for a sample test pattern. Normally, with the exception of primary clocks, input signals can change only at the beginning of each test cycle.

DESIGN FOR TESTABILITY GUIDELINES

The number of primary clocks that can be applied to the Device Under Test (DUT) is a function of the number of timing generators available in the tester. By programming these timing generators, one can select multiple clocks as well as their timing relationship with respect to the test cycle.

The duration of the test cycle varies from system to system and determines the upper limit of frequency at which a device can be tested. Typically, however, there are other constraints that require the testing frequency to be much lower than the capabilities of the tester.

One of the constraints is that all transitions at the outputs of the device must become stable within the same test cycle. Therefore, the maximum propagation delay from input to output of the DUT determines the maximum frequency at which it can be tested.

Another factor influencing the maximum testing frequency is called cycle slip. In some cases, the exact system cycle in which an output signal becomes valid is not critical to the system operation. It is, however, very important to the tester. Testing systems compare the output states to the expected responses on a cycle-by-cycle basis by activating strobe signals (see Figure 6-1) which, in turn, enable a set of comparators connected to the output channels. If output responses were to slip from one test cycle to the next, the device will fail even though it may be fully functional at the system level.

In order to avoid these problems, many semiconductor manufacturers test the functionality of the devices at a relatively low (1 MHz) tester frequency. The dynamic performance of the circuit is verified by selective measurement of propagation delays representative of the overall switching characteristics of the device.

Pin skew within the tester is another phenomenon closely related to cycle slip. Pin skew is defined as the phase shift among the input signals due to the differences in propagation delays within the pin electronics of the tester hardware. If the input signals feeding the data and the clock inputs of a flip flop change at different times, for example, they could cause the circuit to assume the wrong state, and the part will fail.

To prevent this from happening, input signals that must preserve their phase relationships should not be changed on the same test cycle.

DESIGN FOR TESTABILITY GUIDELINES

Test patterns can be greatly compacted by using bursts of clock signals when no other inputs or outputs change state within a test cycle. Figure 6-2 shows an example where clock bursting could be used to minimize the length of a test pattern.

There are cases where outputs might change state when the device is clocked a number of times but the output responses might not be of interest. In those cases the system designer should make the vendor aware of the presence of non-interesting outputs in order to take advantage of the clock bursting technique, thereby allowing compression of the test vector set.

Properly documented test vectors should contain input signals and expected output responses as observed during simulation. For any circuit containing sequential logic, test vectors typically consist of two parts:

1) An initialization step, where all memory or storage elements are set to a known state by exercising the device clock, data, and reset pins.

2) The actual test pattern used to stimulate output pins.

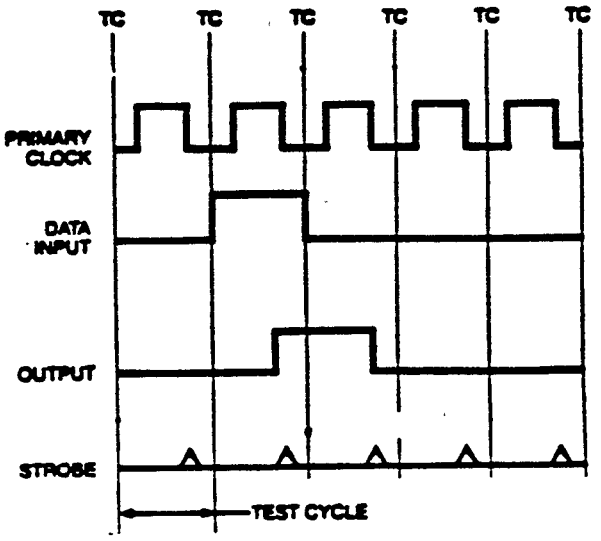
Though test pattern formats may vary among manufacturing companies, Figure 6-3 shows a typical generic test pattern.

Bidirectional pins must have the state of both the input and output signals listed separately. Interpretation of the information will be enhanced if the order of the input pins matches the order of the corresponding output pins. The tester will monitor the pin when the output is enabled, and will drive the pin when it is in the high impedance state.

If a device contains internal three-state busses, the designer must arrange the enable logic so that the busses are always driven (not left in a high impedance state). Furthermore, internal buss contention must be avoided to ensure that the state of the buss propagated to the device output pins during the test sequence will match the expected response from the simulation files.

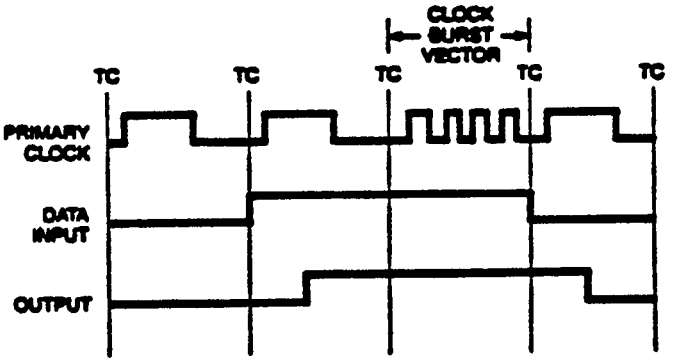
DESIGN FOR TESTABILITY GUIDELINES

Figure 6-1
Sample synchronous pattern for use during functional test



DESIGN FOR TESTABILITY GUIDELINES

Figure 6-2
Example of a clock burst cycle used to compress the length of the functional test pattern



DESIGN FOR TESTABILITY GUIDELINES

Figure 6-3
Sample test pattern format

```
INPUT VECTORS;
SIG0,SIG1,SIG2,SIG3,SIG4,SIG5,SIG6,SIG7,
IDAT1,IDAT2,IDAT3,IDAT4,IDAT5,IDAT6;


OUTPUT SIGNALS;
SUM,CARRY,O0,O1,O2,O3,O4,O5,O6,O7,
ODAT1,ODAT2,ODAT3,ODAT4,ODAT5,ODAT6;
```

INPUTS;	OUTPUTS;
10010101xxxxxx	xxxxxxxxxxxxxx
10100101xxxxxx	xxxx000000001001
10001010xxxxxx	00011000010010001
01010010xxxxxx	10010100101100001
01001010xxxxxx	10010101001100001
10010100xxxxxx	01010010010010010
01001010xxxxxx	01001010010010001
10010010xxxxxx	10011110111101000
11011010xxxxxx	10110110101100000
00001111xxxxxx	00001010000000001
00010100xxxxxx	01001010001010001
11110010xxxxxx	01010111011010000
10100101xxxxx1	10101010101100oz
10010101xxxx01	10101001010000zz
10001010xxx110	10001010010102zzz
10010010xx0101	10010001010022zzz
10000100x10010	10000101001zzzzz
10010101010101	10010010102zzzzz
10010010100101	10010010102zzzzz
10010010010101	00101001002zzzzz
00101000110101	00010100102zzzzz
10001010100011	01001010012zzzzz
10000101011011	00000101012zzzzz
10001010101010	00001010102zzzzz
10001010110101	10000010102zzzzz
10001000010101	00010100102zzzzz
10001000110100	10010001012zzzzz
00100100101011	01010010102zzzzz
10010100101011	01010101102zzzzz
10101100101010	00101010102zzzzz
00101101010101	01010101002zzzzz
00101001000101	10101010102zzzzz

DESIGN FOR TESTABILITY GUIDELINES

CHAPTER 7

ENHANCING DC PARAMETRIC AND BOARD LEVEL TESTING

SYSTEM LEVEL TEST CONSIDERATIONS IN MULTI-CHIP/MULTI-MODULE SEMICUSTOM IC DESIGNS

The discussion in this chapter will be concerned with design methods for assuring the functionality of multiple semi-custom integrated circuits (or Application Specific Integrated Circuits, ASIC's) when they are combined on the circuit board. Although the method presented here is not currently known to be required in any ASIC design system, it is presented as one possible technique to be used by those system designers who want to improve manufacturability.

Two current limitations of production board testers define the requirements of such a design and test methodology. First, because of limited pattern memory, current board testers provide for only a few hundred to a few thousand test vectors. A good fault graded test pattern for one ASIC design may itself be several thousand vectors in length. Second, if an ASIC (or any) component cannot be isolated out of the board environment, it must be back driven by the board tester. This may unduly stress the component. In the case of a CMOS component, back driving may also lead to latchup induced failures due to injected current.

These board tester limitations place several requirements on design and test methodology for ASIC components. First, due to the limited number of vectors available at test, a small (low number of vectors) test for all ASIC components is required. Second, to avoid the stress of backdriving, component isolation of all ASIC components must be provided.

To ensure good fault coverage and yet keep the ASIC component test vector set for circuit board test small, internal fault detection must be performed on the component prior to board test. This important testing methodology should be two-fold. The production test pattern for the component must yield a high fault score for all single stuck-at faults. Using this proven production test pattern, all ASIC components should be tested by the ASIC vendor before shipping to the customer.

DESIGN FOR TESTABILITY GUIDELINES

Assuming that this internal fault detection has already taken place, the size of tests to verify that all I/O cells and connections are good must still be kept small. Utilization of a set of testable I/O cells by all ASIC designs would allow detection of the primary manufacturing failure modes:

* Blown input buffers due to electrostatic discharge

* Blown I/O buffers due to latchup failures

* Bad solder joints and/or socket connections

These cells should also allow all I/O buffers to be placed in a high impedance state, with a minimal number of vectors, to provide component isolation.

A special set of three I/O buffers is required to minimize the number of test vectors required to verify functionality and simplify the dc parametric testing of all input and output buffers. Figure 7-1 shows these three special I/O buffers. Figure 7-2 shows a summary of the I/O Buffer Logic States.

The first of these is a dedicated "-I/O TEST ENABLE" input buffer. If this signal is driven to a logic "0", the state of the output buffers is controlled by signals from the other two special buffers, "+I/O TEST OUTPUT ENABLE" and "+I/O TEST DATA". If this "-I/O TEST ENABLE" signal is driven to a logic "1", or not driven, the state of the output buffers is controlled by the functional data and output enable signals for each buffer.

The second special buffer is a multiplexed "+I/O TEST OUTPUT ENABLE" input buffer. It can be used as a normal input buffer. If "- I/O TEST ENABLE" is active (logic "0"), this input buffer controls the output enable of all of the output buffers. A logic "0" at this input causes all the output buffers to be placed in a high impedance state. A logic "1" causes all of the output buffers to be driven with the value given by the "+ I/O TEST DATA" signal.

The third special buffer is a dedicated "+ I/O TEST DATA" buffer. If "- I/O TEST ENABLE" is inactive (logic "1"), this signal is an output that indicates the Exclusive OR sum of all the inputs except "- I/O TEST ENABLE". If "- I/O TEST ENABLE" is active (logic "0") and + I/O TEST ENABLE" is active (logic "1"), this signal acts as an input that determines whether a logic "1" or "0" will be driven on all of the output buffers.

DESIGN FOR TESTABILITY GUIDELINES

With the above special buffers in place, testing the remaining I/O buffers is simplified. For example, all inputs can be Exclusive OR'd together such that regardless of the state of the other buffers, changing any single output will cause a change of state of the "I/O TEST DATA" output. Therefore, if there are N buffers it will take only (2*N) vectors to verify the functionality of all input buffers. In addition, dc parametric testing of input buffers for Vih and Vil is simplified.

Also, all output buffers can be controlled to either high impedance, logic "1", or logic "0", independent of the internal state of the chip. This means that only three input vectors are required to verify functionality of all output buffers. DC parametric testing of output buffers for Voh versus Ioh and Vol versus Iol is also simplified as well. Figure 7-3 illustrates this testing using the three I/O buffers.

In summary, design and test methodology required to assure system level testability of ASIC's should include the following:

* Special, easily testable three-stateable I/O buffers designed on all ASIC's.

* Vendor's production test pattern must yield a high fault score for all single stuck-at faults.

ON-CHIP TESTABILITY FEATURES

The use of on-chip testability features is intended to allow circuit designers an inexpensive, efficient, and simple method of performing dc parametric tests. In addition, features, such as a ring oscillator, can be added to allow the ac performance of the chip to be monitored. These simple tests are accomplished by dedicating three test pins on each circuit, and placing a multiplexer in front of each output buffer. Figure 7-4 shows a block diagram of the on-chip test features used by one gate array manufacturer. A few points should be noted regarding this diagram. First, the multiplexers in front of the output buffers are built directly into the peripheral structure of the chip. They are not constructed using the core transistors of the gate array. This significantly reduces the propagation delay through the output buffers. A similar approach is taken with the input buffers on the three test pins. Thus, by building the test structures directly into the silicon instead of by using metalization, performance of the device is optimized.

DESIGN FOR TESTABILITY GUIDELINES

Table 7-1 is a truth table detailing the on-chip test functions. There are three dedicated test pins TESTN, TOEN, and TDAT. By setting them all to a logical "1" the chip functions in the normal operation mode. When test enable (TESTN) or test output enable (TOEN) is set to a logical "0" the chip is placed in the test mode. To illustrate this approach, note that when TESTN=TOEN="0" it is possible to test Vol, Voh, Iol, and Ioh for every output buffer in two test cycles. Other parametric tests can be performed in a similar manner.

A ring oscillator has been included to determine the ac performance of the chip. This allows easy and direct access to the part in order to ensure that the process is within expected limits. The ring oscillator is also useful for performing correlation with the manufacturer's data at incoming inspection.

DESIGN FOR TESTABILITY GUIDELINES

Figure 7-1
Special input/output buffers

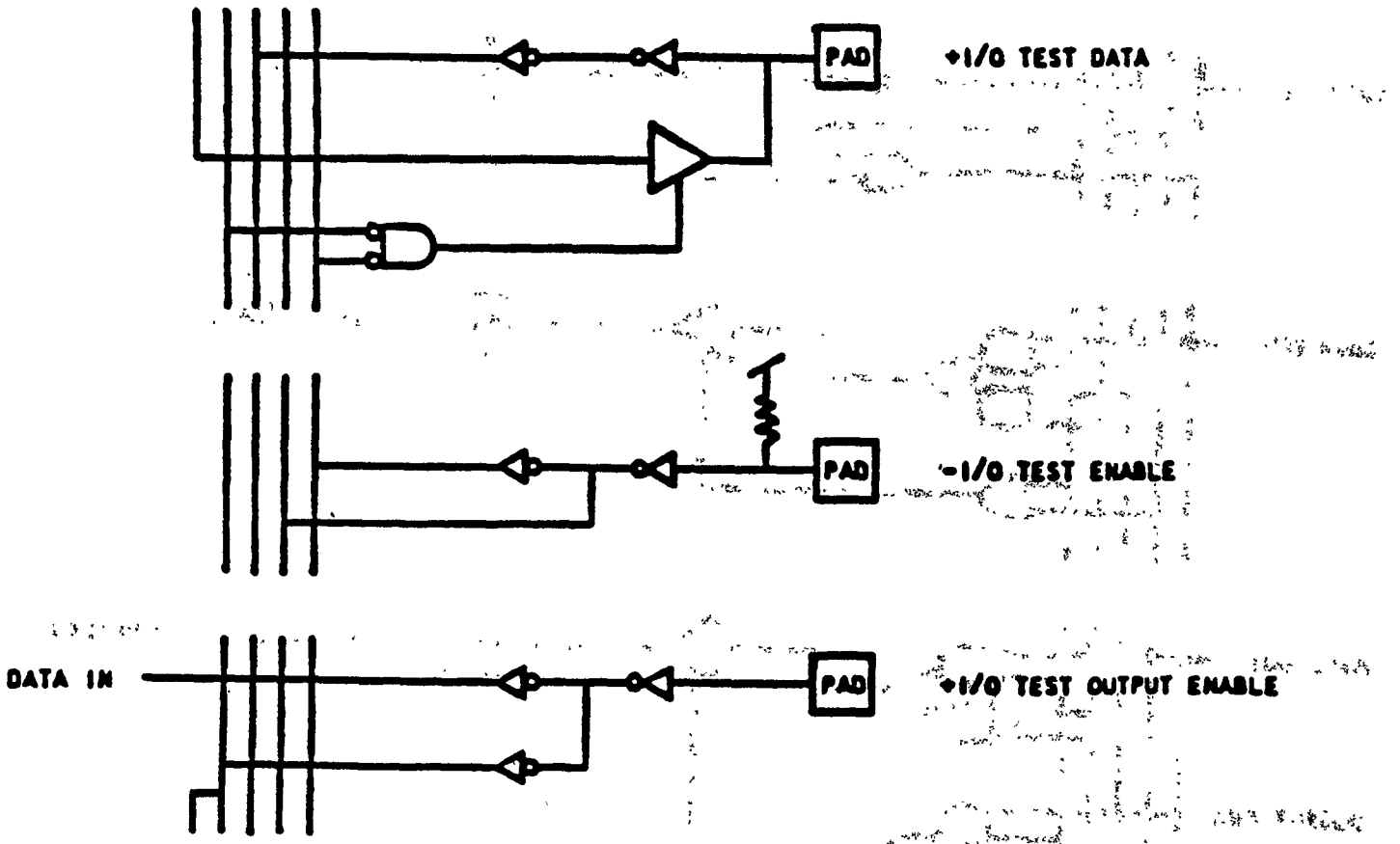
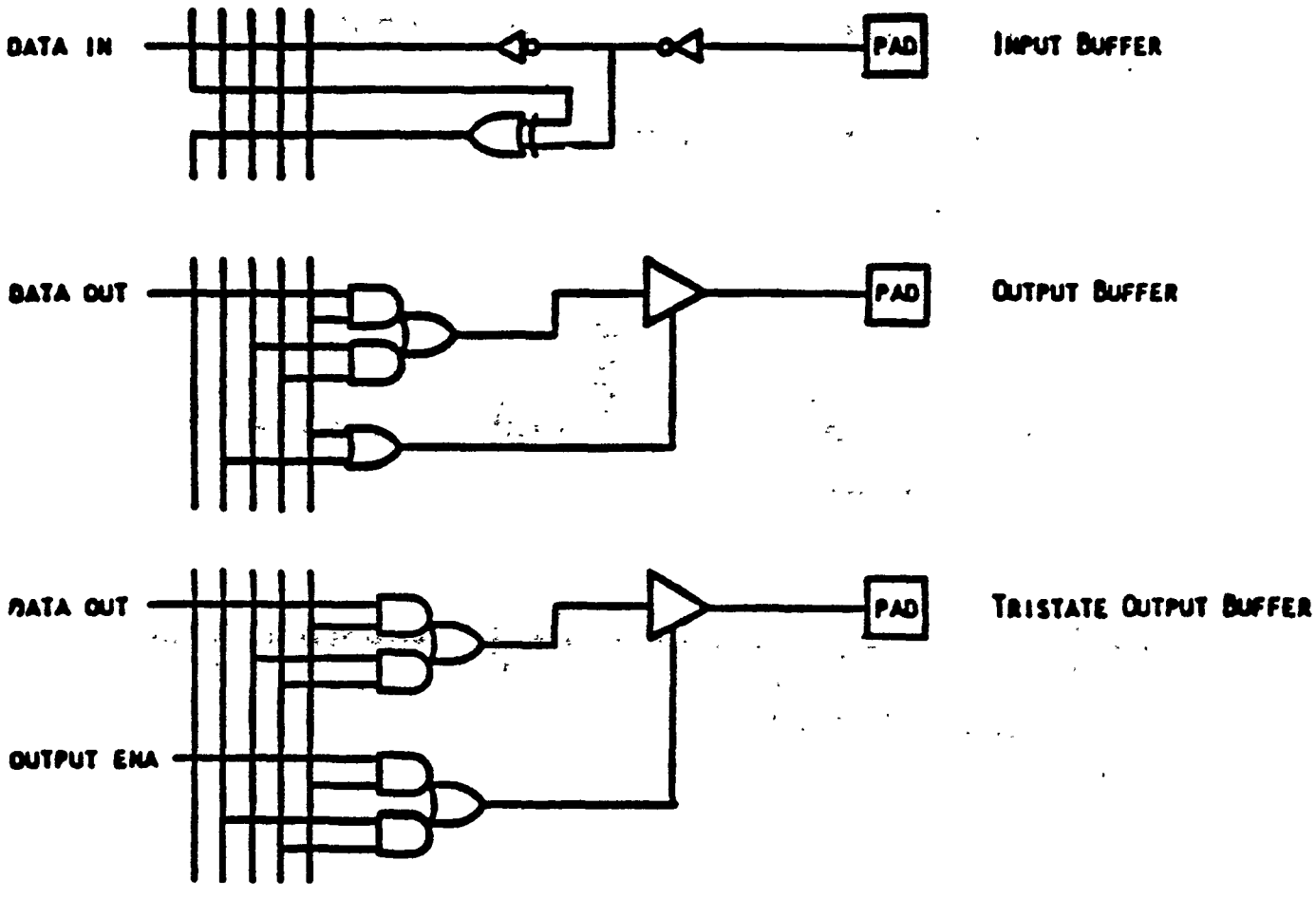


Figure 7-2
I/O Buffer Logic States

-I/O TEST ENAB.	+I/O TEST OUT ENAB.	+I/O TEST DATA	DATA OUT
INPUT: 0	INPUT: 0	HI-Z STATE	HI-Z
	INPUT: 1	INPUT: 0	OUT: 0
		INPUT: 1	OUT: 1
INPUT: 1	PIN USED AS A NORMAL INPUT BUFFER.	PIN USED AS AN OUTPUT. INDICATES EX-OR SUM OF INPUTS.	OUT: NORMAL LOGIC FUNCTION.

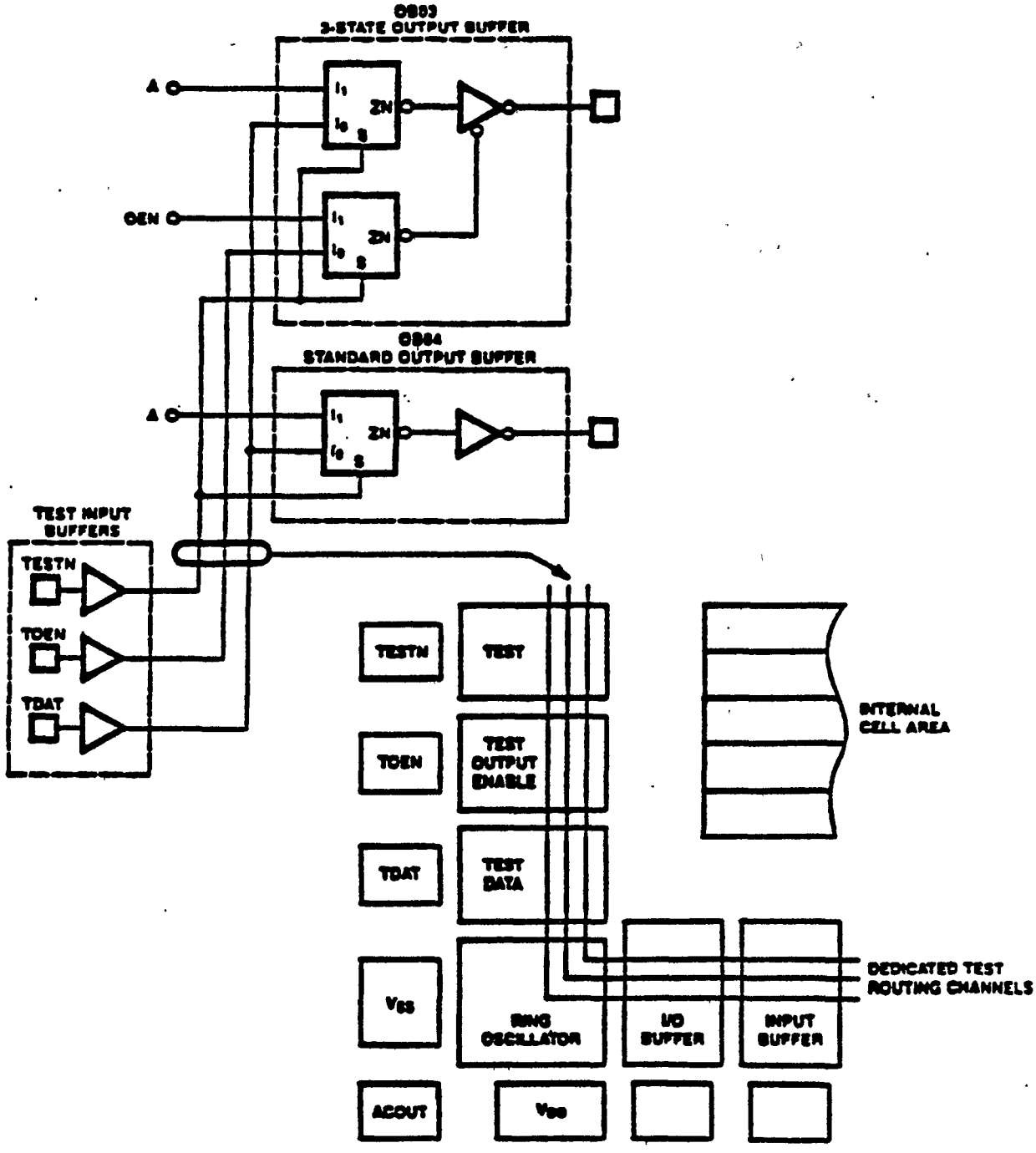
DESIGN FOR TESTABILITY GUIDELINES

Figure 7-3
Testable Input/output Buffer Example



DESIGN FOR TESTABILITY GUIDELINES

Figure 7-4
ON-CHIP TEST DIAGRAM



DESIGN FOR TESTABILITY GUIDELINES

TABLE 7-1
Truth Table for On-chip Testability Features

TEST TYPE	TESTN	TOEN	TDAT	TESTS ACOUT PERFORMED	DESCRIP.
DC PARAMETRIC	0	0	0	DISAB Vol,Iol	TEST DATA
	0	0	1	DISAB Voh,Ioh	ON OUTS
AC MONITOR	0	1	0	ENAB RING OSC	OUTS IN
DC PARAMETRIC	0	1	0	DISAB LEAKAGE	AC MODE
DC PARAMETRIC	0	1	1	DISAB SBY CURRENT	STATIC PWR
USER DEFINED	1	0	0	DISAB	
	1	0	1	DISAB	
	1	1	0	DISAB	
NORMAL	1	1	1	DISAB	NORMAL OP

DESIGN FOR TESTABILITY GUIDELINES

CHAPTER 8

SUMMARY AND CONCLUSION

This document has made an attempt to bring to the reader a brief overview of the necessity for designing with testability as a primary goal. It has also presented short descriptions of some of the more popular methods available with which to achieve this goal.

The application of these methods will vary from company to company and from circuit to circuit. No attempt should be made to suggest that there is only one correct method to use when designing testable circuits. Company policies or contract restrictions may favor particular approaches, and this document is not intended to contradict those policies.

When DFT is maintained as a goal, the designer, his company and the ultimate customer all will be rewarded. The designer will have fewer problems to solve at the time of test. The company will have to spend less time generating test programs and debugging those programs and the circuit. The customer will benefit through improved delivery and shorter incoming inspection cycles.

There is still, however, no such thing as a free lunch. DFT is not without its price. Chip layouts will be more complex. In some cases, they may even operate at slightly lower clock rates. These will be the challenges that will require the judicious use of DFT techniques and good engineering judgment.

As circuits continue to grow in complexity, DFT will move from the status of a design goal to that of an absolute necessity. VLSI and larger circuits with embedded processors and/or memories must be capable of being completely tested if any assurance of quality is to be provided. DFT techniques will continue to improve with time, as will the computer programs that automatically generate test programs to work with those techniques. The future will most certainly be technically challenging (and rewarding) for the circuit design and the design automation tool development communities.

DESIGN FOR TESTABILITY GUIDELINES

(Intentionally left blank)

DESIGN FOR TESTABILITY GUIDELINES

APPENDIX A

BIBLIOGRAPHY OF PUBLISHED LITERATURE

TESTING AND FAULT LOCATION

J. Abadir and Y Deswarte, "Run time program for self checking single board computer" 1982 IEEE International Test Conference , Philadelphia, Nov 15-18, 1982 pp. 205-213.

J. Abraham and K. Parker, "Practical Microprocessor Testing: Open and Closed Loop Approaches," COMPCON Spring 1981 Digest, Feb. 1981.

J. Abraham and H. Shih, "Testing of MOS VLSI Circuits," Proc Intl Symp Circuits and systems, Kyoto, Japan, June 1985

S. B. Akers, "Functional testing with binary decision diagrams," in Proc. 8th Int. Conf. Fault Tolerant Computing, Toulouse, France: IEEE Comput. Soc. June 1978,

G. E. Alderson, N. Benowitz, R. H. Bork, and D. W. Turner, "Advanced avionics fault isolation system (AAFIS)," Vol. 1, AAFIS Concepts Final Rep., Contract N62269-73-C-0132, Hughes Aircraft Co., Culver City, CA., Rep. P73-314, Aug. 1973.

M.A. Annaratone and M.G. Sami, "An Approach to Functional Testing of Microprocessors," Proc. FTCS-82, 1982.

Y. Arzoumanian and J. Waicukauski, "Fault Diagnosis in an LSSD Environment," Digest of Papers, 1981 International Test Conference, Philadelphia, PA, pp. 362-370.

A. Avizienis, "A study of the effectiveness of fault-detecting codes for binary arithmetic," Jet Propulsion Lab. Tech. Rep. 32-711,

P. H. Bardell and W. H. McAnney, "A View from the Trenches: Production Testing of a Family of VLSI Multichip Modules", Proceedings of the Fault-Tolerant Computing Symposium, 1981, pp. 281-283.

DESIGN FOR TESTABILITY GUIDELINES

P. L. Barry, "Failure Diagnosis on the LT1280," IBM Journal of Research and Development, Vol. 27, No. 1, pp. 41-49, Jan. 1983.

Z. Barsilai, J. Savir, G. Markowsky, and M.G. Smith, "The weighted syndrome sums approach to VLSI testing ," IEEE Trans. Comput., Vol. C-30, pp. 996-1000, Dec. 1981.

N. Benowitz, R. H. Bork, D. F. Calhoun, G. W. K. Lee and J. P. Shen, "Advanced avionics fault isolation system (AAFIS) application," Vol. 1, AAFIS Application Final Rep., Contract N62269-73-C-0132/P00001, Hughes Aircraft Co., Culver City, CA., Rep. P74-169, May 1974.

N. Benowitz, J. E. Bauer, and C. T. Joeckel, "Advanced avionics fault isolation system for digital logic," in 1974 ASSC Conf. Rec., San Diego, CA., pp. 195-202, Oct. 30 - Nov. 1, 1974.

N. Benowitz et al, "An Advanced Fault Isolation System for Digital Logic, IEEE Transactions Computers, Vol. C-24,pp 489-497, May 1975.

N. Benowitz, et al., "Fault Detection/Isolation Results from AAFIS Hardware Built-In Test", NAECON '76 Record, pp. 215-222.

N. Benowitz, D.F. Calhoun, G.E. Alderson, J.E. Bauer and C.T. Joeckel, "An Advanced Fault Isolation System for Digital Logic," IEEE Trans on Comp, Vol C-24, No 5, pp 489-497 May 1975

J. M. Berger, " A Note on Error Detection Codes for Asymmetric Channels," Information and Control, Vol. 4, pp. 68-73, March 1961.

I. Berger and Z. Kohavi, "Fault detection in fan-out free combinational networks," IEEE Trans Comput, vol C-22 pp 903-905, Oct 1973

D. K. Bhavsar and R. W. Heckelman, "Self-testing by polynomial division," in the Digest of IEEE Test Conf., pp. 208-216, 1981.

DESIGN FOR TESTABILITY GUIDELINES

T. Blackman, J. Fox, and C. Rosebrugh, "The Silc Silicon Compiler: Language and Features," Proc 22nd Design Automation Conf, Las Vegas, Nev, June 1985, pp 232-237

W.G. Bouricius, E.P. Hsieh, G.R. Putsolu, J.P. Roth, P.R. Schneider, and C.J. Tan, "Algorithms for detection of faults in logic circuits," IEEE trans, Comput., vol. C-20, pp. 1258-1264, Nov 1971.

M. A. Breuer and A. D. Friedman, "Diagnosis and Reliable Design of Digital Systems," Computer Science Press, Inc., 1976.

D. F. Calhoun and G. Wolfe, Jr., "Status and future directions of pad relocation full wafer LSI," in Proc. 23rd Electronics Components Conf., Washington, D. C., May 1973.

J.L. Carter, "The Theory of Signature Testing for VLSI, Proc. 14th ACM Symp. on Theory of Computing, San Francisco, May 1982, pp. 66-76.

W.C. Carter, "Signature Testing with Guaranteed Bounds for Fault Coverage," Dig, 1982, IEEE Test Conf - Cherry Hill 1982, Philadelphia, Penn, pp 75-82, Nov 11-13, 1982

W. C. Carter, "Linear Shift Register Sequence Generators and Systematic Coding of these Sequences", IBM TR00.1282, May, 1965.

W. C. Carter, "The Ubiquitous Party Bit," Dig., Twelfth Annual International Symposium on Fault-Tolerant Computing, (FTCS-12), Santa Monica, CA, pp. 289-296, June 22-24, 1982.

W.C. Carter, "Signature Analysis with Guaranteed Bounds for Fault Coverage, Digest of 1982 International Test Conf., Philadelphia pp75-82, October 1982.

A.Y. Chan, "Easy-to-Use Signature Analyzer Accurately Troubleshoots Complex Logic Circuits." Hewlett-Packard Journal, pp 9-14, May 1977

H. Y. Chang, E.G. Manning and G. Metze, "Fault diagnosis of digital systems" Wiley, New York, 1970

DESIGN FOR TESTABILITY GUIDELINES

H.Y. Chang, E.G. Manning and G. Metze, Fault Diagnosis of Digital Systems" New York, Wiley Interscience, 1970

A. C. L. Chiang and R. McCaskill, "Two new approaches simplify testing of microprocessors," Electronics, pp. 100-105, Jan 22, 1976.

N. Cotsapas and V.K Agarwal, "Analysis of Fault Coverage Masking in Built-in Self-Test Schemes" Proc. 23rd Annual Allerton Conf. on Communication, Control and Computing, Oct 2-4, 1985, pp 183-192

J. J. Curtin and J. Waicukauski, "Multi-Chip Module Test and Diagnostic Methodology," IBM J. Res. Develop. 27, pp. 27-34, 1983.

R. David, "Testing by Feedback Shift Registers," IEEE Trans. Comput., Vol. C-29, pp. 668-673, July 1980.

R. David and G. Blanchet, "About Random Fault Detection in Combinational Networks," IEEE Trans. Compu., Vol. C-25, pp. 659-664, June 1976,

R.David and P. Thevenod-Fosse, "Random Testing of Integrated Circuits", IEEE Trans. Inst. and Meas. Vol. C-25, pp 20-25, March 1981.

R.David, "Feedback shift register testing", Digest 8th Annual Int. Conf. Fault-Tolerant Computing, pp 103-107, Toulouse, June 1978

R.David, "Testing by feedback shift register", IEEE Trans. on Computers, vol. C-29, pp 668-673, July 1980.

R. David, "Signature Analysis of Multi Output Circuits", Proc.. 14th Intl. Symposium on Fault Tolerant Computing, June 1984, pp 366-371

R. P. Davidson, M. L. Harrison, and R. L. Wadsack, "BELLMAC-32, A Testable 32-Bit Microprocessor", Proceedings of the International Test Conference, pp.15-20, 1981.

G. H. deVisme, "Binary Sequences", The English Universities Press. Ltd., London, 1971.

DESIGN FOR TESTABILITY GUIDELINES

F. J. O. Dias, "Truth-table verification of an iterative logic array." IEEE Trans. Comput. Vol. C-25, pp. 605-613, June 1976.

E. B. Eichelberger, "Hazard Detection in Combinational and Sequential Circuits IBM J. Res. Develop., 9 (1965), pp. 99.

E. B. Eichelberger, "Level Sensitive Logic System," U. S. Patent 37832? January 1, 1974.

E. B. Eichelberger, "Method of Propagation Delay Testing a Functional Logic System," U. S. Patent No. 3??, January 8, 1974.

P. P. Fasang, "A Fault Detection and Isolation Technique for Microcomputers," 1982 IEEE International Test Conference 214-219 Nov 1982

W. G. Fee - Tutorial "LSI Testing", 2nd ed. IEEE Computer Society, IEEE Catalog No. EHO 122-2, 1978.

A.D. Friedman and P.R. Menon, "Fault Detection in Digital Circuits," Englewood Cliffs NJ, Prentice Hall 1971

R.A.Frohwerk," Signature Analysis: A new Digital Field Service Method," Hewlett-Packard Journal, vol. 28, no. 5,pp 2-8, May 1977.

H. Fujiwara and K. Kinoshita, "Testing logic circuits with compressed data," in Proc. 8th Annu. Int. Conf. on Fault Tolerant Comput., June 1978, pp. 108-113.

G. J. Gerbier and J. P. L. Berger, "Binary Tester for Logic Circuit Subassemblies," U. S. Patent 3286175, Dec. 6, 1963.

S. W. Golomb, Shift Register Sequences, Aegean Park Press, Laguna Hills, Calif., Revised Edition, 1982.

G. Gordon and Hans Nadig, "Hexadecimal Signatures Identify Troublespots in Microprocessor Systems", Electronics, March 1977, pp. 89-96.

DESIGN FOR TESTABILITY GUIDELINES

S. Z. Hassan, D. J. Lu, and E. J. McCluskey, "Parallel Signature Analyzers -- Detection Capability an Extensions," Dig., COMPCON Spring 83, San Francisco, CA, Feb 28 - Mar 3, 1983.

S. Z. Hassan, "Algebraic Analysis of Parallel Signature Analyzers," Center for Reliable Computing Technical Report No. 82-5, Computer Systems Laboratory, Stanford University, Stanford, CA, June 1982.

S. Z. Hassan, and E. J. McCluskey, "Testing PLA's Using Multiple Parallel Signature Analyzers", Dig Thirteenth Annual International Symposium on Fault-Tolerant Computing. (FTCS-13), Milan, Italy, pp 422-425, June 28-30, 1982

J. P. Hayes, "Testing Logic Circuits by Transition Counting," Digest of Papers: 1975, International Symposium on Fault Tolerant Computing, Paris, pp. 215-219, June 1975.

J. P. Hayes, "Check Sum Test Methods", FTCS-6, June 1976, pp. 114-119.

J. P. Hayes, "Check Sum Methods for Test Data Compression," Journal of Design Automation and Fault Tolerant Computing, pp. 3-17, June 1976.

J. P. Hayes, "Transition count testing of combinational logic circuits," IEEE Trans. Comput., Vol. C-25, pp. 613-620, June 1976.

Hewlett-Packard, "A designers guide to signature analysis," Application Note 222, Palo Alto, Calif, April 1977

T. Higuchi and M. Kameyama, "Ternary logic system based on T gate", in Proc 5th Intl Symp Multiple Valued Logic, May 1975 pp 290-304

E. R. Holland and J. L. Robertson, "GUEST-A Signature-Analysis-Based Test System for ECL Logic," Hewlett-Packard J. 33, pp. 26-29, 1982.

S. J. Hong and D. L. Ostapko, "FITPLA: A Programmable Logic Array for Functional Independent Testing," Digest FCTS10, 10th Annual Conference on Fault- Tolerant Computing, Kyoto, Japan, 1980, pp. 131-136.

M. Hu and K. C. Smith, "On the use of CMOS ternary gates to realize a self-checking binary logic system," in Proc 11th Intl Symp Multiple-Valued Logic, May 10981, pp 212-217

K. Hua et al, "Built in Tests for VLSI Finite State Machines," Digest of Papers FTCS 14, Kissimmee, Fla, June 1984 pp 292-297

O.H. Ibara and S. K. Sahni, "Polynomially complete fault detection problems," IEEE Trans. Comput., Vol. C-24, pp. 242-249, March 1975.

M. Ichikawa, "Constant Weight Code Generators," CRC Technical Report No 82-7 Center for Reliable Computing, Stanford University, June 1982

T. Jackson, P. Vais and K. Schwerbock, "A new approach to on board microprocessor-based self test," 1982 international Test Conference, Philadelphia, Nov 15-18, 1982, pp 537-540

I. Kohavi and Z. Kohavi, "Detection of multiple faults in combinational logic networks," IEEE Trans Comput vol C-21 pp 556-568 June 1972.

A. Krasniewski and A. Albicki, "Simulation Free Estimation of Speed Degradation in NMOS Self-Testing Circuits for CAD Applications," Proc 22nd Design Automation Conf, Las Vegas, Nev, June 1985, pp. 808-811

J. Losq, "Referenceless random testing," in Proc 6th Int. Symp, Fault Tolerant Computing, pp. 108-113, Pittsburgh, Pa, June 1976

J. Losq, "Efficiency of Random Compact Testing," IEEE Trans. Comput., Vol. C-27, pp. 516-525, June 1978.

DESIGN FOR TESTABILITY GUIDELINES

N. P. Lyons, "Faultrack: universal fault isolation procedure for digital logic," 1974 IEEE Intercon Tech Progam, New York, March 1974, paper no. 40/2.

E. Manning, "On computer self-diagnosis: Part I and Part II" IEEE Trans. Electron. Comput., Vol. EC-15. pp. 873-890. Dec 1966.

G. Markowsky, "Syndrome-Testability Can Be Achieved by Circuit Modification", IEEE TC, Aug. 1981, pp. 604-606.

J. E. Meggitt, "Error correcting codes and their implementation for data transmission systems, "IRE Trans. Inform. Theory, Vol. IT-7, pp. 234-244, Oct. 1961.

M. R. Mercer and V. D. Agrawal, "A novel clocking technique for VLSI circuit testability, "IEEE J. Solid-State Circuits, vol SC-19 pp 207-212, Apr 1984

J. Mucha, "Hardware techniques for testing VLSI circuits based on built in test," Proc. COMPCON '81, pp. 366-369, Feb. 1981.

J. C. Munzio, and D. M. Miller, "Spectral techniques for fault detection," Proc. 12th Ann. Fault-Tolerant Computing Symp., pp. 297-302, June 1982.

H. J. Nadig, "Testing a Microprocessor Product Using Signature Analysis", Proc. 1978 Semiconductor Test Conference, Oct. 31 - Nov. 2, 1978, Cherry Hill, pp. 159-169.

K. P. Parker, "Compact testing: Testing with compressed data," In Proc. 6th Int Symp Fault-Tolerant Computing, pp. 93-98 Pittsburgh, PA, June 1976

W. W. Peterson, "Encoding and Error-Correction Procedures for the Bose- Chauduri Codes", IRE Trans. Inform. Theory IT-6, 1960, pp. 459- 470. W. W. Peterson and D. T. Brown, "Cyclic codes for error detection," Proc. IRE, Vol. 49, pp. 228-235, Jan 1961.

W. W. Peterson and E. J. Weldon, Jr., Error-Correcting Codes. The MIT Press, Second Edition, Dec. 1975.

R. L. Pierson and T. B. Williams, "The LT1280 for Through-the-Pins Testing of the Thermal Conduction Module," IBM Journal of Research and Development, Vol. 27, No. 1, pp. 35-40, Jan. 1983.

R. A. Rasmussen, "Automated Testing of LSI," Computer Digest pp.69-78

J. C. Rault, "A graph theoretical and probablistic approach to the fault detection of digital circuits," in Proc. 1st Int. Symp. Fault- Tolerant Computing, IEEE Computer Society Pub. 71C-G.C., pp. 26-29, Mar. 1971

R. Rhodes-Burke, "Applying signature analysis to existing processor-based products," Electronics, vol 54 no 4 pp 127-133, Feb 24 1981

C. Robach and G. Saucier, "Diversified test methods for local control units," IEEE Trans. Comput., Vol. C-24, pp. 562-567, May 1975.

C. Robach and G. Saucier, "Dynamic Testing of Control Units," IEEE Trans. on Comput., Vol. C-27, pp. 617-623, July 1978.

R. L. Russo, "On the Tradeoff Between Logic Performance and Circuit-To- Pin Ratio for LSI," IEEE Trans. Comput., Vol. C-21, No. 2, pp. 147-153, February 1972.

S. C. Russel, "Incoming inspection and test programs," Electronics Test, pp 46-57, Oct 1980

S. H. Sangani, et al., "In-Situ Testing of Combinational and Memory Circuits Using a Compact Tester", Proc. 8th Annual International Conference on Fault-Tolerant Computing, June 21-28, 1978, Toulouse, p. 214.

J. Savir, "Detection of Single Intermittent Faults in Sequential Circuits," IEEE Trans. Comput., Vol. C-29, pp. 673-678, July 1980.

DESIGN FOR TESTABILITY GUIDELINES

J. Savir, G. Ditlow, and P. H. Bardell, "Random Pattern Testability," Dig., Thirteenth Annual International Symposium on Fault Tolerant Computing, (FTCS-13), Milan Italy, pp. 80-89, June 28-30, 1982.

J. Savir and P. H. Bardell, "On Random Pattern Test Length," IBM Technical Report TR 00.3224, Poughkeepsie, New York, June 1983.

M. Serra and J. C. Muzio, "Testing Programmable Logic Arrays by Sum of Syndromes," IEEE Workshop on Design for Testability, Beaver Creek, Colo, April 1985

F. F. Sellers, M. Y. Hsiao, and L. W. Fearnson, "Analyzing Errors with Boolean Difference", IEEE TC, July 1968, pp. 676-683.

S. C. Seth, "Data Compression techniques in logic testing: an extension of transition counts," Journal of Design Auto and Fault Tolerant Computing, vol 1, pp 99-114, Feb 1977

J.J. Shedlestky, "A rationale, for the random testing of combinational digital circuits," in Proc. CompCon 1975 Fall, pp 5-8, Sept 1975

J. J. Shedletsky, "Random Testing: Practicality vs. Verified Effectiveness," Dig., Seventh Annual International Conference on Fault-Tolerant Computing, (FTCS-7), Los Angeles, CA, pp. 175-179, June 28-30.

J. E. Smith, "Measures of the Effectiveness of Fault Signature Analysis," IEEE Trans.Comput., Vol. C-29, No. 6, pp. 510-514, June 1980.

J. E. Smith, "The prospects for multivalued logic: A technology and applications view," IEEE Trans. Comput Vol C-30, pp 619-634, Sept 1981

I. H., Spector, "Combining Logic State and Signature Analysis", Application Note of Paratronics, Inc., San Jose, California.

DESIGN FOR TESTABILITY GUIDELINES

T. Sridhar and J. P. Hayes. "Testing bit-sliced microprocessors," in Proc. 9th. Int. Conf. Fault-Tolerant Computing. Madison, WI: IEEE Comput. Soc., June 1979. pp. 211-218.

T. Sridhar and J. P. Hayes, "A Functional Approach to Testing Bit-Sliced Microprocessors," IEEE Trans. on Comput., Vol. C-30, pp. 563-571, August 1981.

T. Sridhar and D. S. Ho., and T. J. Powell, and S. M. Thatte, "Analysis and Simulation of Parallel Signature Analyzers," Dig., 1982, IEEE Test Conf., - Cherry Hill '82, Philadelphia, Penn., pp. 656-661, Nov. 11-13, 1982.

T. Sridhar and J. P. Hayes, "Self testing bit sliced microprocessors," IEEE Comp Society 22nd Int'l Conf COMPCON Spring 1981, San Francisco, Feb. 23-26, 1981, pp 312-316

J. H. Stewart, "Application of Scan/Set for Error Detection and Diagnostic", Proceedings of the International Test Conference, pp. 152-158, 1978.

S. Strom, "ROM-resident software self tests microcomputers," Electronic Design, Vol 28, No 13, 131-132, 134, June 21, 1980

A.K. Susskind, "Testing by verifying walsh coefficients, "Proc. 11th Ann. Fault-Tolerant Computing Symp., pp. 206-208, June 1981.

R. Tellez-Giron, R. David, "Random Fault Detection in Logical Networks" Digest IFAC International Symposium on Discrete Systems, Riga, USSR, Oct 1974

S. M. Thatte and J.A. Abraham, "A methodology for functional level testing of microprocessors," in Proc. 8th Int. Conf. Fault-Tolerant Computing. Toulouse, France: IEEE Comput. Soc., June 1978, pp. 90-95.

S. M. Thatte and J. A. Abraham, "User Testing of Microprocessors," COMPCON Spring 1979, San Francisco, pp. 108-114, 1979.

DESIGN FOR TESTABILITY GUIDELINES

P. Thevenod-Fosse and R. David, "A Method to Analyze Random Testing of Sequential Circuits," Digital Processes, Vol. 4, pp. 313-332, 1978.

P. Thevenod-Fosse and R. David, "Random Testing of the Data Processing Section a Microprocessor", Proceedings of the Fault-Tolerant Computing Symposium, pp. 275-280, 1981.

P. Thevenod-Fosse and R. David, "Random testing of the control section of a microprocessor," FTCS-83, pp. 366-373.

C. Timoc, F. Stott, K. Wickman, L. Hess, "Adaptive Self-Test for a Microprocessor", Proceedings of the International Test Conference, pp. 701-703, 1983.

A. Toth and C. Holt, "Automated Data Base Drive Digital Testing," Computer (January 1974), pp. 13-19. Tzidon, A., I. Berger, and M. Yoeli, "A Practical Approach to Fault Detection in Combinational Networks," IEEE Trans. on Comp., Vol.C-27, No. 10, pp. 968-971, Oct. 1978.

C.S. Venkatraman and K.K. Saluja, "Transition count testing of sequential machines," Digest 10th Fault-Tolerant Computing symp, Kyoto, pp 167-172, Oct 1980

J. Wakerly, Error Detecting Codes, Self Checking Circuits and Applications, Elsevier North-Holland, 1978.

E. White, "Signature analysis - enhancing the serviceability of microprocessor based industrial products," Proc IECI, pp 68-76, March 1978

DESIGN FOR TESTABILITY GUIDELINES

FAULTS AND FAULT MODELING

P. Banerjee and J. Abraham, "Fault characterization of VLSI MOS circuits," IEEE International Conference on circuits and computers, ICCC-82, pp. 564-568.

R. Boute, "Algebraic properties of test sequences and fault relations," Tech Rep 37, Digital Systems Laboratory, Stanford University, Stanford, CA, Nov 1972

K. K. Chakrabarti, A. K. Choudhury, & M. S. Basu, "Complementary Function Approach to the Synthesis of Three-Level NAND Network", IEEE Trans. Comp., C19, 6, June, 1970, 509-514.

R. Chandramouli, "On testing stuck-open faults,"' Proc. FTCS-83, pp. 258-265.

R. Chandramouli and H. Sucar, "Defect Analysis and Fault Modeling in MOS Technology", Proc. 1985 International Test Conference, Nov 1985, pp 313-321

Y. El-Ziq, "Classifying, Testing, and Eliminating VLSI MOS failures," VLSI Design, Sept 1983, pp 30-35.

J. Gailiay, Y. Crouzeet, and M. Vergniault, "Physical Versus Logical Fault Models MOS LSI Circuits: Impact on their Testability," IEEE Trans. on Computers, Vol, C-29, No. 6, June 1980.

J. P. Hayes, "A NAND Model for Fault Diagnosis in Combinational Logic Networks," IEEE TC, Dec. 1971, pp. 1496-1506.

J. P. Hayes, "On Modifying Logic Networks to Improve their Diagnosability", IEEE TC, Nov. 1972, pp. 1183-1188.

I. Koren, "Analysis of the signal reliability measure and an evaluation procedure," IEEE Trans. Comput., Vol. C-28, pp. 244-249, March 1979.

L. Mattera, "Component reliability, Part I: Failure data bears watching," Electronics, pp. 91-98, Oct 1975

K. C. Y. Mei, "Bridging and Stuck-At Faults", IEEE Trans. on Comput., Vol. C-23, No. 7, pp. 720-727, July 1974.

K. P. Parker, and E. J. McCluskey, "Analysis of logic circuits with faults using input signal probabilities," IEEE Trans. Comput., Vol. C-24, pp. 573-578, May 1975.

K.P. Parker, and E. J. McCluskey, "Probabilistic treatment of combinational networks," IEEE Trans. Comput., Vol. C-24, pp. 668-670, June 1975.

J. Savir, "Testing for Single Intermittent Failures in Combinational Circuits by Maximizing the Probability of Fault Detection," IEEE Trans. Comput., Vol. C-29, pp. 410-416, May 1980.

J. J. Shedletsky, and E. J. McCluskey, "The Error Latency of a Fault in Combinational Digital Circuits," Proc 5th Ann Fault-Tolerant Computing Symp pp 210-214, June 1975

J. J. Shedlestky and E. J. McCluskey, "The Error Latency of a Fault in a Sequential Digital Circuit," IEEE Trans. Comput., Vol. C-25, pp. 655-659, June 1976.

K. C. Y. Mei, "Bridging and Stuck-At Faults", IEEE Tran. on Comput., Vol. C-23, No. 7, pp. 720-727, July 1974.

R. L. Wadsack, "Technology Dependent Logic Faults", Proceedings of COMPCON S'78, pp. 124-129, 1978.

R. L. Wadsack, "Fault modelling and Logic Simulation of CMOS and MOS Integrated Circuits," The Bell System Technical Journal, Vol. 57, No. 5, May-June 1978.

R. L . Wadsack, "How much fault coverage is enough?" In dig 1981 Int Test Conf pp 547-554, Oct 1981

T. W. Williams and N. C. Brown, "Defect Level as a Function of Fault Coverage", IEEE TC, Dec. 1981. pp. 987-982.

DESIGN FOR TESTABILITY GUIDELINES

DESIGN FOR TESTABILITY

J. Abadir and M. Breuer, "A Knowledge Based System for Designing Testable VLSI chips," IEEE Design & Test of Computers, Aug 1985, pp 56-68

J. A. Abraham and D. D. Gajski, "Design of testable structures defined by simple loops, "IEEE Transactions on Computers, vol C-30 No 11, Nov 1981, pp 875-884

V. D. Agrawal, et al, "Automation in Design for Testability," Proc Custom Integrated Circuits Conf, Rochester, NY, May 1984, pp 159-163

D. A. Anderson, "Design of self-checking digital networks," Coordinated Sci. Lab. Rep. R-527, University of Illinois, Urbana, Sept. 1971.

H. Ando, "Testing VLSI with Random Access Scan" Proceedings of COMPCON Symp 80, pp. 50-52, 1980.

P. H. Bardell - Organizer, "Built-In Test--Theory and Implementations" IEEE Computer Society Workshop 1985, Tutorial Notes

P. H. Bardell, and W. H. McAnney, "Self-Testing of Multichip Logic Modules", Proceedings of the International Test Conference, pp. 200-204, 1982. F. P. M. Beenker, "Systematic and Structured Methods for Digital Board Testing", Proceedings of the 1985 International Test Conference, pp. 380-385,

R. G. Bennets, "Design of Testable Logic Circuits", Addison Wesley, 1984

N.C. Berglund, March 1979. "Level-Sensitive Scan Design Tests Chips, Boards, System," Electronics.

F. Beucler and B. Manner, "HILDO: The Highly Integrated Logic Device Observer", VLSI Design, June 1984, pp 88-96

DESIGN FOR TESTABILITY GUIDELINES

S. Bozorgui-Nesbat, and E.J. McCluskey, "Structured Design for Testability to eliminate Test Pattern Generation," Digest, 10th Annual Symposium on Fault-Tolerant Computing (FTCS-10), Kyoto, Japan, pp 158-163, Oct 1-3, 1980

M. A. Breuer and X. Zhu, "A knowledge Based system for selecting a test methodology for a PLA," Proc 22nd Design Automation Conf, Las Vegas, Nev, June 1985, pp 259-265

M. A. Breuer, and A. Friedman, "Design to Simplify Testing," in Diagnosis and Reliability of Digital Systems, Computer Science Press, Inc. Woodland Hills, Calif, pp 291-303, 1976

M. A. Breuer, "Automatic Design of Testable Circuits", ATPG Workshop Proc., San Francisco, CA mar 1983, pp 3-6

M. A. Breuer, "Automatic Design for Testability Based on a Cost Measure," Proc. Autotestcon, Long Island, N.Y., Nov. 1983, pp 138-143

H. H. Butt and Y.M. Elziq, "Impact of Mixed-Mode Self Test on Life Cycle Cost of VLSI Based Boards", Proc. 1984 International Test Conf., Oct. 1984 pp. 338-347

W. C. Carter et al., "Design of serviceability Features for the System/360", IBM J. of R&D, Mar. 1964, pp. 115-126.?

C. W. Cha, "A Testing Strategy for PLAs", Proceedings of the Design Automation Conference, pp. 326-334, 1978.

T. H. Chen and M. A. Breuer, "Automatic Design for Testability via Testability Measures," IEEE Trans. Computer Aided Design, Vol CAD-4, No 1, Jan 1985, pp 3-11

W. Daehn and J. Mucha, "A Hardware Approach to Self-Testing of Large Programmable Logic Arrays," IEEE Trans. Computers C-30, pp. 829-833, (1981).

DESIGN FOR TESTABILITY GUIDELINES

S. DasGupta, R. G. Walther, T. W. Williams, and E. B. Eichelberger, "An Enhancement to LSSD and Some Applications of LSSD in Reliability, Availability, and Serviceability", Proceedings of the Fault- Tolerant Computing Symposium, pp. 32-34, 1980.

S. DasGupta, P. Goel, R. G. Walther, and T. W. Williams, "A Variation of LSSD and its Implications on Design and Test Pattern Generation in VLSI", Proceedings of the International Test Conference, pp. 63-66, 1982.

E. B. Eichelberger, "Method of Level Sensitive Testing a Functional Logic System," U. S. Patent 3761695, Sept 25, 1973.

E. B. Eichelberger & T. W. Williams, "A Logic Design Structure for LSI Testability", Jour. Design Auto. & Fault-Tol. Comp., 2, May, 1978, 165-178.

E. B. Eichelberger, and T. W. Williams, "A logic design structure for LSI testability," Proc. 14th Design Automation Conference, pp. 462-468, June 1977.

E. B. Eichelberger and E. Lindbloom, "Random-Pattern Coverage Enhancement and Diagnosis for LSSD Logic Self Test," IBM Journal of Research and Development, Vol. 27, No. 3, pp. 265-272, May 1983.

H. Eiki, K. Inagaki, and S. Yajima, "Autonomous Testing and its Application to Testable Design of Logic Circuits," Digest FTCS-10, Annual International Conference on Fault-Tolerant Computing, Kyoto, Japan, 1980, pp. 173-179.

Y. El-Ziq and H.H. Butt, "A Mixed-Mode Built-In Self-Test Using Scan Path and Signature Analysis" Proc 1983 International Test Conf, Oct. 1983 pp 269-272

P. P. Fasang, "BIDCO, built-in digital circuit observer, "Proc. 1980 Int. Test Conference, pp. 261-266, Nov. 1980.

DESIGN FOR TESTABILITY GUIDELINES

A.D. Friedman, "Easily Testable Iterative Systems," IEEE trans Computers, Vol C-22, Dec 1973, pp 1061-1064

H. Fujiwara and S. Toida, "The complexity of fault detection: an approach to design for testability," Proc. 12th Ann. Fault-Tolerant Computing Symp., pp. 101-108, June 1982.

S. Funatsu, N. Wakatsuki, and A. Yamada, "Designing Digital Circuits with Easy Testable Consideration", Proceedings of the International Test Conference, pp. 98-102, 1978.

H. S. Fung and S. Hirschhorn "Optimizing solutions before freezing designs: automated design for testability," Design & Test of Computers Feb 1986

H. S. Fung, S. Hirschhorn, and R. Kulkarni, "Design for Testability in a Silicon Compilation Environment," Proc 22nd Design Automation Conf Las Vegas, Nev June 1985, pp 190-196

H. S. Fung, "A testable by Constructions Strategy for the Silc Silicon Compiler," Proc Intl Conf Computer Design: VLSI in Computers Port Chester NY, Oct 1985, pp 554-557

G. Grassl and H. J. Pfleiderer, "A Self-Testing PLA", Proceedings of the International Solid-State Circuits Conference, pp. 60-61, 1982.

J. P. Hayes, and E. J. McCluskey, "Testability Considerations in Microprocessor-Based Design," Computer, pp 17-26, March 1980

P. Horstmann, "Design for Testability Using Logic Programming," Intl Test Conf Proc, Philadelphia, Pa, Oct 1983, pp 706-713

P. Horstmann, "Automation of the Design for Testability Using Logic Programming," PhD dissertation, Syracuse University, 1983 M. Hu and K. C. Smith, "Ternary Scan Design for VLSI Testability", IEEE Transactions on Computers, Feb 1986, Volume C-35, No 2.

DESIGN FOR TESTABILITY GUIDELINES

D. Komonytsky, November, 1982. "LSI Self-Testing Using Level-Sensitive Scan Design and Signature Analysis," Digest and Signature Analysis," Digest of Papers, 1982 International Test Conference, Philadelphia, PA, pp. 414-424.

D. Komonytsky, "Synthesis of Techniques Creates Complete System Self-Test," Electronics, Vol. 26, No. 5, pp. 110-115, March 1983.

B. Konemann, J. Mucha, and G. Zwiehoff, "Built In Logic Block Observation techniques," Proc. 1979 Test Conference, pp. 37-41, Oct. 1979.

B. Konemann, J. Mucha, and G. Zweihoff, "Built-in test for complex digital integrated circuits," IEEE Journal of Solid State Circuits, Vol. SC-15, No. 3, pp. 315-319, June 1980.

Kuban and Salick, "Testability Features of the MC68020" Proc. 1984 Intl Test Conf, Oct. 1984, pp 821-826

G. P. Mak, J. A. Abraham and E. S. Davidson, "The Design of PLAs With Concurrent Error Detection," Proc. FTCS-12, Los Angeles, CA, pp. 303-310, June 1982.

E. J. McCluskey and S. Bozorgui-Nesbat, "Design for Autonomous Test," IEEE Trans on Computers Vol C-30 No 11 pp 866-875, nov 1981

E. J. McCluskey, "Built-In Verification Test," Dig., 1982 IEEE Test Conf. - Cherry Hill '82, Philadelphia, Penn., pp. 183-190, Nov. 11-13, 1982.

K. L. Meinert, "Designing for testability with GUEST," Hewlett-Packard., P. 28, March 1982.

R. K. Montoye and J. A. Abraham, "Built in tests for arbitrarily structured carry-lookahead VLSI adders," Proc VLSI 83, Int'l Conf on Very Large Scale integration, Trondheim, Norway August 16-19 1983,

DESIGN FOR TESTABILITY GUIDELINES

E. I. Muehldorf, "Designing LSI Logic for Testability," 1976 Semiconductor Test Symp., October 19-21, 1976 IEEE Computer Soc. and Philadelphia Section of the IEEE, pp. 45-49.

C. C. Perkins, S. Sangani, H. Stopper, and W. Valitski, "Design for In- Situ Chip Testing with a Compact Tester," Dig., 1980, IEEE Test Conf., Philadelphia, Penn., Nov. 11-13, 1980.

D. R. Resnick, "Testability and Maintainability with a New 6K Gate Array," VLSI Design, Vol. 4, No. 2, pp. 34-39, March/April 1983

J. Savir, "Syndrome-Testable Design for Combinational Circuits", IEEE Trans. Comp., C29, 6, June, 1980, 442-451.

J. Savir, "Syndrome-testable design of combinational circuits," IEEE Trans. Comp., Vol. C-29, pp. 442-451, June 1980. See also "Correction to 'Syndrome-testable design of combinational circuits,'" ibid, pp. 1012-13, Nov. 1980.

J. Savir, "Syndrome-Testing of 'Syndrome-Untestable' Combinational Circuits," IEEE Trans. on Comp., Vol. C-30, No. 8, pp. 606-608, 1981.

R. M. Sedmak, "Design for Self-Verification: An Approach for Dealing with Testability Problems in VLSI-Based Designs," Digest of Papers, 1979 IEEE Test Conference, Cherry Hill, NJ, pp. 112-120.

R. Sedmak, "Built-in Self-Test: Pass or Fail?", IEEE Design & Test Vol 2, April 1985, pp 17-19

M. T. M. Segers, "A Self-Test Method for Digital Circuits," Proc. 1981 IEEE Test Conf., pp. 79-85, October 1981.

T. Sridhar and J. P. Hayes, Design of easily testable bit-sliced systems," IEEE Trans Comput vol C-30 pp 842-854

DESIGN FOR TESTABILITY GUIDELINES

L. A. Stolte and N. C. Berglund, "Design for Testability of the IBM System/ 38", Proceedings of the International Test Conference, pp. 29-36, 1979.

S. M. Thatte, "VLSI design for testability," IEEE Comput Soc Workshop on Fault Tolerant VLSI design, Santa Monica, California, April 1980 Reported in IEEE Computer, vol 12, p 53, Dec 1980

S. M. Thatte, D. S. Ho, H.-T. Yuan, T. Sridhar, and T.J. Powell, "An architecture for testable VLSI processors, "Proceedings of the International Test Conference, pp.484-492, 1982.

C. Timoc, J. M. Favennec, and C. Le Blanche, "A Testable Regular Design", Proceedings of the International Conference on Circuits and Computers, pp. 210-213, 1982

J. Turino, Design for Testability, Campbell, CA Logic Solutions Inc. 1979

R. Treuer, H. Fujiwara, and V. K. Agarwal, "Implementing a Built In Self Test PLA Design," IEEE Design & Test of Computers, Apr. 1985 pp 37-49

E. Trischler, "Design for Testability Using Incomplete Scan Path and Testability Analysis," Siemens Forsch-U Entwickl.Ber Bd, Vol 13 No 2, 1984 Springer-Verlag, pp 56-61

M. J. Y. Williams and J. B. Angell, "Enhancing Testability of Large- Scale Integrated Circuits via Test Points and Additional Logic", IEEE Trans. on Computers, pp. 46-60, January 1973.

T. Williams., and K. Parker. January 1982. "Design for Testability - A Survey," IEEE Transactions on Computers, Vol. C31, No. 1.

T. W. Williams and E. B. Eichelberger, "Random Patterns within a Structured Sequential Logic Design", Proceedings of the Intl. Test Conference, pp. 19-26, 1977.

T. W. Williams and K. P. Parker, "Testing logic networks and designing for testability," Computer pp 9-21 Oct 1979

DESIGN FOR TESTABILITY GUIDELINES

FAULT SIMULATION AND TESTABILITY ANALYSIS

M. Abramovici et al. "Critical Path Tracing: An Alternative to Fault Simulation", IEEE Design & Test, Feb. 1984, pp 83-92

V. D. Agrawal and M. R. Mercer, "Testability Measures-What Do They Tell Us ?" Digest of Papers, 1982 International Test Conference, Philadelphia, PA, pp. 362-370.

V. D. Agrawal, S. C. Seth, and P. Agrawal, "Fault coverage requirement in production testing of LSI circuits" IEEE J. Solid State Circuits vol SC-17 pp 57-61, Feb 1982

E. Archambeau, "Testability Analysis Techniques: A Critical Survey", VLSI Systems Design, pp. 46-52, Dec. 1985.

D. B. Armstrong, " A Deductive Method for Simulating Faults in Logic Circuits", IEEE Trans. on Computers, Vol. C-21, No. 5, pp.464-471, May 1972.

A. K. Bose, P. Kozak,C-Y Lo, H. N. Nham, E. Pacas-Skewes, and K. Wu, "A Fault Simulator for MOS LSI Circuits", Proceedings of the Design Automation Conference, pp. 400-409, 1982.

H. Y. Chang, S. G. Chappell, C. H. Elmendorf, L. D. Schmidt, "Comparison of Parallel and Deductive Fault Simulation Methods", IEEE Trans. on Computers, Volume C-233, No. 11, pp. 1132-1138, November 1974.

M. M. Denneau, "The Yorktown Simulation Engine", Proceedings of the 19th Design Automation Conference, pp. 55-59, 1982.

J. Fong, "On Functional Controllability/Observability Analysis" Proc IEEE Int'l Test Conf, Philadelphia, Pa, nov 1982 pp 170-175

J. Fong, "A Generalized Testability Analysis Algorithm for Digital Logic Systems," Proc Int'l Symp Circuits and Systems, Rome Italy, May 1982

H. S. Fung and J. Fong, "An Information Flow Approach to Functional Testability Measures," Proc Int'l Conf Circuits and Computers, New York, NY Sept 1982, pp 460-463

L. H. Goldstein, "Controllability/Observability Analysis of Digital Circuits," IEEE Trans. Circuit Systems CAS-26, pp. 685-693, 1979.

L. H. Goldstein and E. L. Thigpen, "SCOAP: Sandia Controllability/ Observability Analysis Program," Proc. 17th Ann. Design Automation Conference, pp. 190-196, June 1980.

J. Grason, "TMEAS - A Testability Measurement Program," Proc. 16th Ann. Design Automation Conference, pp. 156-161, June 1979.

S. Jain and V. D. Agrawal, "STAFAN: An Alternative to Fault Simulation" Proc. 21st Design Auto. Conf., June 1984, pp 18-23

K. P. Parker, "Software Simulator Speeds Digital Board Test Generation," Hewlett-Packard Journal, pp. 13-19, March 1979.

W. Rogers and J. Abraham, "High Level Hierarchical Fault Simulation Techniques," Proc ACM Spring Computer Science Conf New Orleans, La, Mar 1985, pp 89-97

W. Rogers and J. Abraham, "CHIEFS: A Concurrent, Hierarchical and Extensible Fault Simulator", Proc ITC 1985, Nov. 1985, pp 710-716

J. Savir, "Good controllability and observability do not guarantee good testability," IBM Research Report, RC 9432, June 1982.

J. E. Stephenson and J. Grason, "A testability measure for register transfer level digital circuits," in Proc Int Fault Tolerant Comput Symp June 21-23 1976 pp 101-107 K. Son, "Fault Simulation with the Parallel Value List Algorithm", VLSI Systems Design, pp. 36-43, Dec. 1985.

E. W. Thompson and S. A. Szygenda, "Parallel Fault Simulation" Computer, pp. 38-44, March 1975.

E. G. Ulrich and T. Baker, "The Concurrent Simulation of Nearly Identical Digital Networks", Proceedings of the Design Automation Workshop, pp. 145-150, 1973.

J. A. Waicukaiski et al, "Fault Simulation for Structured VLSI", VLSI Systems Design, pp. 20-32, Dec. 1985.

TEST GENERATION

J. A. Abraham and S. M. Thatte, "Fault Coverage of Test Programs for a Microprocessor," 1979 IEEE International Test Conference, pp.18-22, Oct. 1979.

M. Abramovici et al. "Test Generation In Lamp2: System Overview", Proc. 1985 Intl. Test Conf., Nov 1985, pp 45-48

M. Abramovici et al. "Test Generation In Lamp2: Concepts and Algorithms", Proc. 1985 Intl. Test Conf., Nov 1985, pp 45-48

K.K. Aggarwal, "Output probability expression for general combinational networks," Microelectron. Reliab., Vol. 17, No. 6, pp. 601-602, 1978.

P. Agrawal, and V. D. Agrawal, "Probabilistic analysis of random test generation methods for irredundant combinational logic networks," IEEE Trans. Comput., Vol. C-24, pp. 691-695. July 1975.

V. D. Agrawal and P. Agrawal, "An Automatic Test Generation System for Illiac IV Logic Boards", IEEE Trans. on Comput., Volume C-21. No. 9, pp. 1015-1017, September 1972.

DESIGN FOR TESTABILITY GUIDELINES

D. B. Armstrong, "On Finding a nearly minimal set of fault detection tests for combinational logic nets," IEEE Trans Electron Comput vol EC-15 pp 73 Feb 1966

P. S. Bottorff, R.E. France, N. H. Garges, and E. J. Orosz, "Test Generation for Large Logic Networks", Proceedings of the Design Automation Conference, pp. 479-485, 1977.

M. A. Breuer, "A random and an algorithmic technique for fault detection and test generation for sequential circuits," IEEE Trans. Compt., Vol. C-20, pp.1364-1370, Nov. 1971.

D. F. Calhoun, L. I. Corrigan, and G. Fox, "A philosophy for and experience with automated test generation for digital modules," presented at the IEEE Region Six Conf., Sacramento,CA, Paper 4A-2, May 11-13, 1971.

H. Fujiwara and K. Kinoshita, "A Design of Programmable Logic Arrays with Universal Tests," IEEE Trans Computers, Vol C-30, Nov 1981, pp 823-828

H. Fujiwara, K. Kinoshita, and H. Ozaki, "Universal Test Sets for Programmable Logic Arrays". Proceedings of the International Fault- Tolerant Computing Symposium, pp. 137-142, 1980.

H. Fujiwara and T. Shimono, "On the Acceleration of Test Generation Algorithms", IEEE Trans. Comp. Vol. C-32, Dec. 1983, pp 1137-1144

S. Funatsu, N. Wakatsuki, and T. Arima, "Test Generation System in Japan," Proc. 12th Ann. Design Automation Conf., Boston (June 1975), pp. 23-25.

P. Goel, "Test generation cost analysis and projections," Proc. 17th Design Automation Conference, pp. 79-84, June 1980.

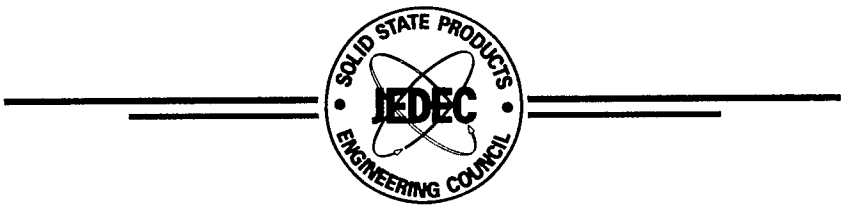
P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits", IEEE Trans. Comp. vol C-30, March 1981, pp 215-222.

DESIGN FOR TESTABILITY GUIDELINES

J. P. Hayes, "Generation of optimal transition count tests," IEEE Trans. Computers, vol C-27, pp 36-41, Jan 1978

J. P. Hayes, "On realization of Boolean functions requiring a minimal or near minimal number of tests", IEEE Trans Comput vol C-20 pp 1506-1513 Dec 1971 M. Y. Hsiao, "Generating PN sequences in parallel," Proc. 3rd Annual Princeton Conf. Information Sciences and Systems, March 1969.

S. Noujaim et at., "A Structured Approach To Test Vector Generation," Proc Intl Conf Computer Design, Port Chester, NY, Oct 1984, pp 757-762


SOLID STATE PRODUCTS
JEDEC
ENGINEERING COUNCIL