4. FAULT MODELING

About This Chapter

First we discuss the representation of physical faults by logical faults and the major concepts in logical fault modeling — namely explicit versus implicit faults, structural versus functional faults, permanent versus intermittent faults, and single versus multiple faults. We introduce the main types of structural faults — shorts and opens — and show how they are mapped into stuck and bridging faults. Next we deal with fault detection and redundancy, and the fault relations of equivalence and dominance. Then we discuss the single and the multiple stuck-fault models, and we present extensions of the stuck-fault model for modeling faults in RTL models. Bridging faults, functional faults, and technology-specific faults are treated in separate chapters.

4.1 Logical Fault Models

Logical faults represent the effect of physical faults on the behavior of the modeled system. (For studies of physical faults in integrated circuits, see [Case 1976, Partridge 1980, Timoc *et al.* 1983, Banerjee and Abraham 1984, Shen *et al.* 1985].) Since in modeling elements we differentiate between the logic function and timing, we also distinguish between **faults that affect the logic function** and **delay faults** that affect the operating speed of the system. In this chapter we will be mainly concerned with the former category.

What do we gain by modeling physical faults as logical faults? First, the problem of fault analysis becomes a logical rather than a physical problem; also its complexity is greatly reduced since many different physical faults may be modeled by the same logical fault. Second, some logical fault models are technology-independent in the sense that the same fault model is applicable to many technologies. Hence, testing and diagnosis methods developed for such a fault model remain valid despite changes in technology. And third, tests derived for logical faults may be used for physical faults whose effect on circuit behavior is not completely understood or is too complex to be analyzed [Hayes 1977].

A logical fault model can be explicit or implicit. An **explicit fault model** defines a fault universe in which every fault is individually identified and hence the faults to be analyzed can be explicitly enumerated. An explicit fault model is practical to the extent that the size of its fault universe is not prohibitively large. An **implicit fault model** defines a fault universe by collectively identifying the faults of interest — typically by defining their characterizing properties.

Given a logical fault and a model of a system, we should be able in principle to determine the logic function of the system in the presence of the fault. Thus, fault modeling is closely related to the type of modeling used for the system. Faults defined in conjunction with a structural model are referred to as **structural faults**; their effect is to modify the interconnections among components. **Functional faults** are defined in conjunction with a functional model; for example, the effect of a functional fault may be to change the truth table of a component or to inhibit an RTL operation.

Although intermittent and transient faults occur often, their modeling [Breuer 1973, Savir 1980] requires statistical data on their probability of occurrence. These data are needed to determine how many times an off-line testing experiment should be repeated to maximize the probability of detecting a fault that is only sometimes present in the circuit under test. Unfortunately, this type of data is usually not available. Intermittent and transient faults are better dealt with by on-line testing. In this chapter we will discuss only **permanent faults**.

Unless explicitly stated otherwise, we will always assume that we have at most one logical fault in the system. This simplifying **single-fault assumption** is justified by the *frequent testing strategy*, which states that we should test a system often enough so that the probability of more than one fault developing between two consecutive testing experiments is sufficiently small. Thus if maintenance intervals for a working system are too long, we are likely to encounter multiple faults. There are situations, however, in which frequent testing is not sufficient to avoid the occurrence of multiple faults. First, what may appear in the real system between two consecutive testing experiments is a physical fault, and some physical faults manifest themselves as multiple logical faults. This is especially true in high-density circuits, where many physical faults can affect an area containing several components. Second, in newly manufactured systems prior to their first testing, multiple faults are likely to exist. And third, if the testing experiment does not detect every single fault (which is usually the case), then the circuit may contain one of the undetected faults at any time, and the occurrence of a second single fault between two testing experiments creates a multiple fault. But even when multiple faults are present, the tests derived under the single-fault assumption are usually applicable for the detection of multiple faults, because, *in most cases, a multiple fault can be detected by the tests designed for the individual single faults that compose the multiple one.*

In general, *structural fault models assume that components are fault-free and only their interconnections are affected.* Typical faults affecting interconnections are shorts and opens. A **short** is formed by connecting points not intended to be connected, while an **open** results from the breaking of a connection. For example, in many technologies, a short between ground or power and a signal line can make the signal remain at a fixed voltage level. The corresponding logical fault consists of the signal being **stuck at** a fixed logic value v ($v \in \{0,1\}$), and it is denoted by *s-a-v*. A short between two signal lines usually creates a new logic function. The logical fault representing such a short is referred to as a **bridging fault**. According to the function introduced by a short we distinguish between AND bridging faults and OR bridging faults.

In many technologies, the effect of an open on a unidirectional signal line with only one fanout is to make the input that has become unconnected due to the open assume a constant logic value and hence appear as a stuck fault (see Figure 4.1(a)). This effect may also result from a physical fault internal to the component driving the line, and without probing the two endpoints of the line we cannot distinguish between the two cases. This distinction, however, is not needed in edge-pin testing, where we can assume that the entire signal line is stuck. Note how a single logical fault, namely the line i stuck at $a \in \{0,1\}$, can represent many totally different physical faults: i open, i shorted to power or ground, and any internal fault in the component driving i that keeps i at the logic value a.

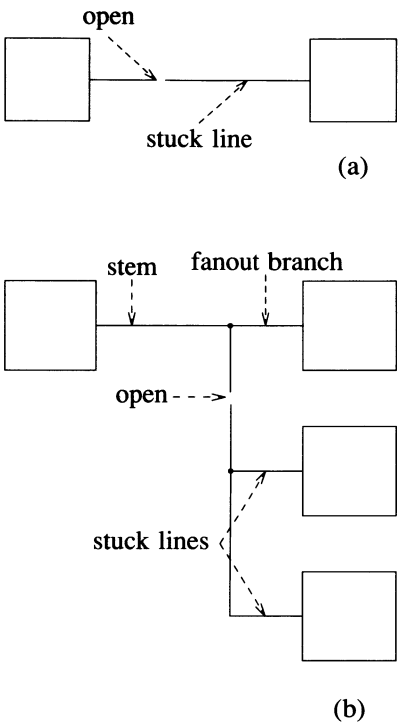


Figure 4.1 Stuck faults caused by opens (a) Single stuck fault (b) Multiple stuck fault

An open in a signal line with fanout may result in a multiple stuck fault involving a subset of its fanout branches, as illustrated in Figure 4.1(b). If we restrict ourselves to the single stuck-fault model, then we have to consider any single fanout branch stuck fault separately from the stem fault.

In the macro approach for hierarchical modeling, every component is expanded to its internal structural model. However, if components are individually tested before their assembly, then it may be enough to test only for faults affecting their interconnections. Then we do not want to consider the faults internal to components but only faults associated with their I/O pins. This restricted fault assumption is referred to as the *pin-fault model.*

4.2 Fault Detection and Redundancy

4.2.1 Combinational Circuits

Let $Z(x)$ be the logic function of a combinational circuit N, where x represents an arbitrary input vector and $Z(x)$ denotes the mapping realized by N. We will denote by t a specific input vector, and by $Z(t)$ the response of N to t. For a multiple-output

circuit $Z(t)$ is also a vector. The presence of a fault f transforms N into a new circuit N_f. Here we assume that N_f is a combinational circuit with function $Z_f(x)$. The circuit is tested by applying a sequence T of test vectors $t_1, t_2, ..., t_m$, and by comparing the obtained output response with the (expected) output response of N, $Z(t_1), Z(t_2), ..., Z(t_m)$.

Definition 4.1: A test (vector) t **detects** a fault f iff $Z_f(t) \neq Z(t)$.

Note that the tests in the sequence T may be applied in any order; therefore, for a combinational circuit we will refer to T as a *set of tests*. Note that Definition 4.1 does not apply if N_f is a sequential circuit. We also emphasize that this definition of detection assumes edge-pin testing with full comparison of the results; other definitions apply for other forms of testing, such as compact testing techniques, which will be described in a separate chapter.

Example 4.1: In the circuit of Figure 4.2, let f be the OR bridging fault between x_1 and x_2. This fault changes the functions realized by the two outputs to $Z_{1f} = x_1 + x_2$ (instead of $Z_1 = x_1x_2$) and to $Z_{2f} = (x_1 + x_2)x_3$ (instead of $Z_2 = x_2x_3$). The test 011* detects f because $Z(011) = 01$ while $Z_f(011) = 11$. □

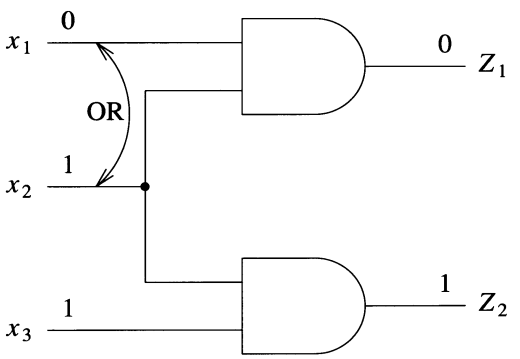


Figure 4.2

For a single-output circuit, a test t that detects a fault f makes $Z(t)=0$ and $Z_f(t)=1$ or vice versa. Thus, the set of all tests that detect f is given by the solutions of the equation

$$Z(x) \oplus Z_f(x) = 1 \tag{4.1}$$

where the symbol $\oplus$ denotes the exclusive-OR operation.

---

* The order of the bits is always $x_1, x_2, ..., x_n$ (or alphabetical order if the names $A, B, C...$ are used).

Example 4.2: The function realized by the circuit of Figure 4.3(a) is $Z = (x_2+x_3)x_1 + \bar{x}_1x_4$. Let f be x_4 s-a-0. In the presence of f the function becomes $Z_f = (x_2+x_3)x_1$, and equation (4.1) reduces to $\bar{x}_1x_4 = 1$. Thus any test in which $x_1 = 0$ and $x_4 = 1$ is a test for f. The expression $\bar{x}_1x_4$ represents, in compact form, any of the four tests (0001, 0011, 0101, 0111) that detect f. □

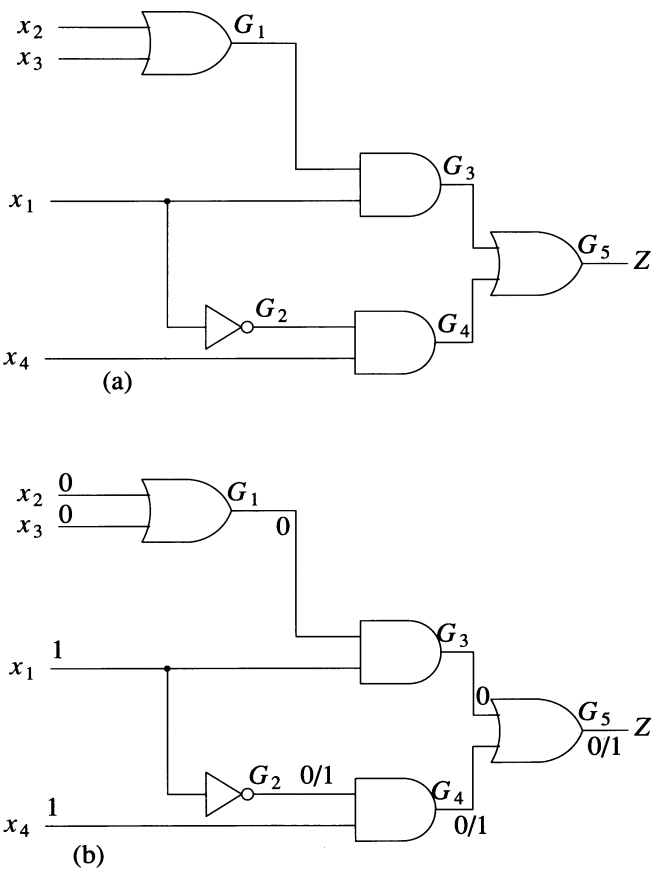


Figure 4.3

Sensitization

Let us simulate the circuit of Figure 4.3(a) for the test $t = 1001$, both without and with the fault G_2 s-a-1 present. The results of these two simulations are shown in Figure 4.3(b). The results that are different in the two cases have the form v/v_f, where v and v_f are corresponding signal values in the fault-free and in the faulty circuit. The fault is detected since the output values in the two cases are different.

Figure 4.3(b) illustrates two basic concepts in fault detection. First, a test t that detects a fault f **activates** f, i.e., **generates an error** (or a **fault effect**) by creating different v and v_f values at the site of the fault. Second, t **propagates the error** to a primary output w, that is, makes all the lines along at least one path between the fault site and w have different v and v_f values. In Figure 4.3(b) the error propagates along the path (G_2, G_4, G_5). (Sometimes the term "fault propagation" is used instead of "error propagation" or "fault-effect propagation.") A line whose value in the test t changes in the presence of the fault f is said to be **sensitized** *to the fault f by the test t.* A path composed of sensitized lines is called a **sensitized path.**

A gate whose output is sensitized to a fault f has at least one of its inputs sensitized to f as well. The following lemma summarizes the properties of such a gate.

Lemma 4.1: Let G be a gate with inversion i and controlling value c, whose output is sensitized to a fault f (by a test t).

1. All inputs of G sensitized to f have the same value (say, a).
2. All inputs of G not sensitized to f (if any) have value $\overline{c}$.
3. The output of G has value $a \oplus i$.

Proof

1. Let us assume, by contradiction, that there are two sensitized inputs of G, k, and l, which have different values. Then one of them (say, k) has the controlling value of the gate. In the presence of the fault f, both k and l change value and then l has the controlling value. But this means that G has the same value independent of f, which contradicts the assumption that the gate output is sensitized to f. Therefore all the sensitized inputs of G must have the same value (say, a).
2. The output of G cannot change in the presence of f if one of its inputs that is not sensitized to f has value c. Hence all inputs of G not sensitized to f (if any) must have value $\overline{c}$.
3. If $a = c$, the gate output has value $c \oplus i$. If $a = \overline{c}$, then all inputs of G have value $\overline{c}$, and the gate output has value $\overline{c} \oplus i$. Hence, in both cases the gate output has value $a \oplus i$. □

The value $\overline{c}$ is referred to as an *enabling value*, since it enables error propagation. Thus a NAND gate that satisfies Lemma 4.1 has either all inputs (sensitized or not) set to 1 or all sensitized inputs set to 0 and the remaining inputs (if any) set to the enabling value 1 (see Figure 4.4).

By repeatedly applying Part 3 of the above lemma, we can derive the following properties of sensitized paths.

Corollary 4.1: Let j be a line sensitized to the fault l *s-a-v* (by a test t), and let p be the inversion parity of a sensitized path between l and j.

1. The value of j in t is $\overline{v} \oplus p$.

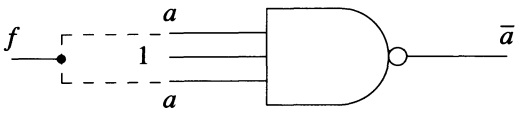


Figure 4.4 NAND gate satisfying Lemma 4.1 (c=0, i=1)

2. If there are several sensitized paths between l and j, then all of them have the same inversion parity. □

Detectability

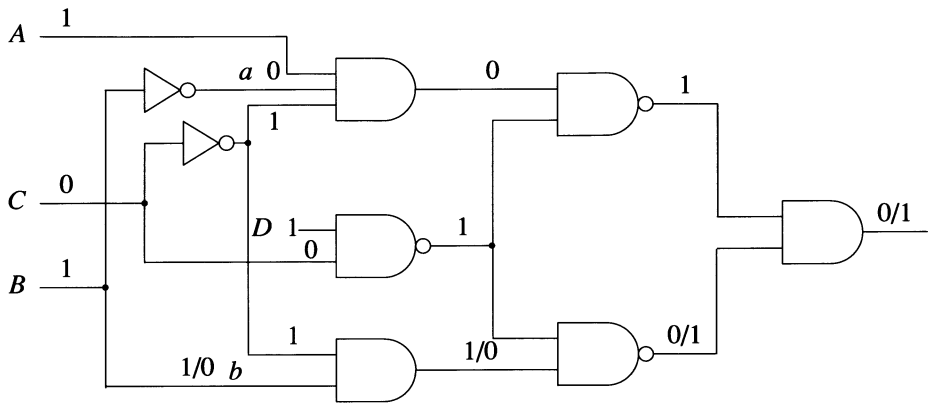
A fault f is said to be **detectable** if there exists a test t that detects f; otherwise, f is an *undetectable* fault. For an undetectable fault f, $Z_f(x) = Z(x)$ and no test can simultaneously activate f and create a sensitized path to a primary output. In the circuit of Figure 4.5(a) the fault a *s-a*-1 is undetectable. Since undetectable faults do not change the function of the circuit, it may appear that they are harmless and hence can be ignored. However, a circuit with an undetectable fault may invalidate the single-fault assumption. Recall that based on the frequent testing strategy, we assume that we can detect one fault before a second one occurs. But this may not be possible if the first fault is undetectable.

When generating a set of tests for a circuit, a typical goal is to produce a *complete detection test set*, that is, a set of tests that detect any detectable fault. However, a complete test set may not be sufficient to detect all detectable faults if an undetectable one is present in the circuit [Friedman 1967].

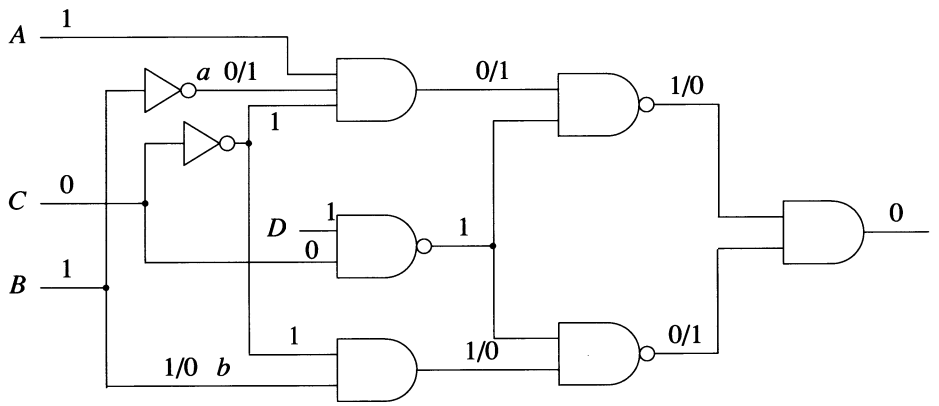
Example 4.3: Figure 4.5(a) shows how the fault b *s-a*-0 is detected by t = 1101. Figure 4.5(b) shows that b *s-a*-0 is no longer detected by the test t if the undetectable fault a *s-a*-1 is also present. Thus, if t is the only test that detects b s-a-0 in a complete detection test set T, then T is no longer complete in the presence of a *s-a*-1.□

The situation in which the presence of an undetectable fault prevents the detection of another fault by a certain test is not limited to faults of the same category; for example, an undetectable bridging fault can similarly invalidate a complete test set for stuck faults.

Example 4.4: [Kodandapani and Pradhan 1980]. Consider the circuit of Figure 4.6(a) that realizes the function $xy + \bar{x}z$. The OR bridging fault between y and $\bar{x}$ is undetectable, since the function realized in the presence of the fault is $xy + yz + z\bar{x} = xy + z\bar{x}$. Figure 4.6 shows how the test 111 detects the fault q *s-a*-0 but no longer does so in the presence of the bridging fault. The test set T = {111, 010, 001, 101} is a complete detection test set for single stuck faults, and 111 is the only test in T that detects q *s-a*-0. Hence T is no longer complete in the presence of the undetectable bridging fault. □



(a)



(b)

Figure 4.5

Redundancy

A combinational circuit that contains an undetectable stuck fault is said to be **redundant**, since such a circuit can always be simplified by removing at least one gate or gate input. For instance, suppose that a *s-a-1* fault on an input of an AND gate G is undetectable. Since the function of the circuit does not change in the presence of the fault, we can permanently place a 1 value on that input. But an n-input AND with a constant 1 value on one input is logically equivalent to the $(n-1)$-input AND obtained by removing the gate input with the constant signal. Similarly, if an AND input *s-a-0* is undetectable, the AND gate can be removed and replaced by a 0 signal. Since now

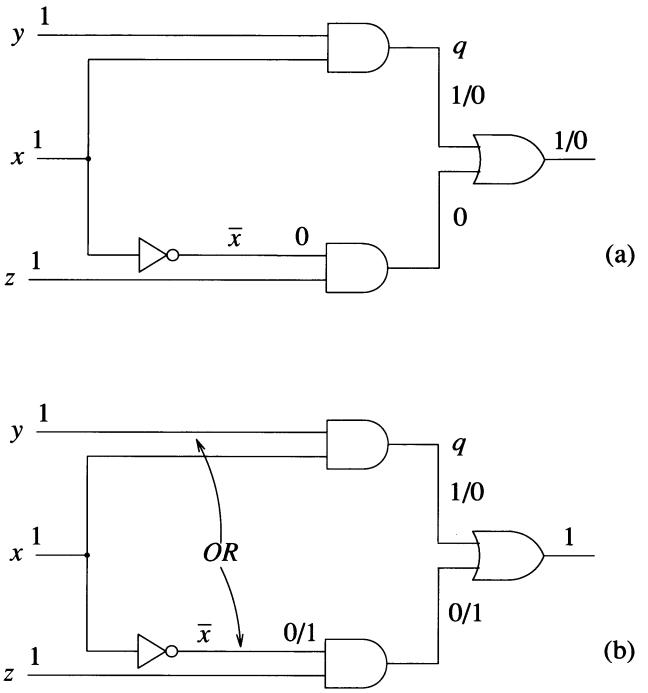


Figure 4.6

we have a new signal with a constant value, the simplification process may continue. The simplification rules are summarized in the following table.

Undetectable fault	Simplification rule
AND(NAND) input *s-a-1*	Remove input
AND(NAND) input *s-a-0*	Remove gate, replace by 0(1)
OR(NOR) input *s-a-0*	Remove input
OR(NOR) input *s-a-1*	Remove gate, replace by 1(0)

The general concept of redundancy is broader than the particular one related to the existence of undetectable stuck faults, and it denotes a circuit that can be simplified. One possible simplification, not covered by the rules given above, is to replace a string of two inverters by a single line. A general type of redundancy [Hayes 1976] exists in a circuit when it is possible to cut a set of r lines and to connect $q \leq r$ of the cut lines to some other signals in the circuit without changing its function. In the following, we will restrict ourselves to the definition of redundancy related to undetectable stuck faults. The term "redundant" is also applied to undetectable stuck faults and to the lines that can be removed. A combinational circuit in which all stuck faults are detectable is said to be *irredundant.*

Redundancy does not necessarily denote an inefficient or undesirable implementation of a function. For example, triple modular redundancy (TMR) is a basic technique used in fault-tolerant design. For the TMR configuration shown in Figure 4.7, any fault occurring in one of the identical modules A will be masked by the two other fault-free modules because of the majority voter circuit M. But this also makes the TMR circuit almost untestable by off-line testing. The solution is to remove the redundancy for off-line testing.

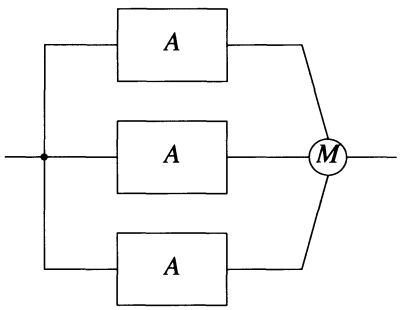


Figure 4.7 TMR configuration

Redundancy may also be introduced in a circuit to avoid hazards. This design technique is illustrated by the circuit of Figure 4.8, which implements the function $z = ab + bc + \bar{a}c = ab + \bar{a}c$. Thus the gate Y that produces the term bc is not logically required. Without this gate, however, the circuit would have a static hazard, as a spurious 0-pulse may appear at the output when the input vector changes from 111 to 011. The role of gate Y is to keep the output constant at 1 during this transition. But the fault Y s-a-0 is undetectable.*

We noted how the presence of a redundant fault may invalidate a complete test set. Other problems that can arise in redundant circuits include the following [Friedman 1967]:

1. If f is a detectable fault and g is an undetectable fault, then f may become undetectable in the presence of g (see Problem 4.6). Such a fault f is called a *second-generation redundant fault.*
2. Two undetectable single faults f and g may become detectable if simultaneously present in the circuit (see Problem 4.7). In other words, the multiple fault $\{f,g\}$ may be detectable even if its single-fault components are not.

---

* The fault Y s-a-0 could be detected by observing the spurious 0-pulse that the term bc is used to prevent. However, we assume a testing experiment in which only the static output values are observed. Moreover, the occurrence of the spurious pulse depends on the delays of the circuit.

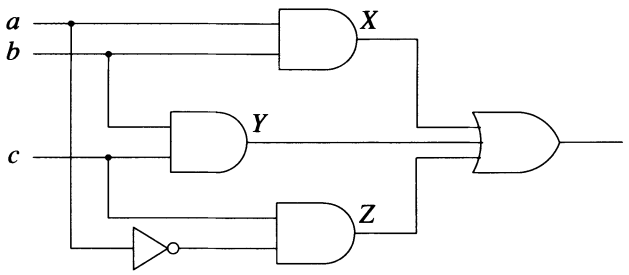


Figure 4.8

Note that in practice, when we deal with large combinational circuits, even irredundant circuits may not be tested with complete detection test sets. The reason is that generating tests for some faults may consume too much time, and all practical test generation programs are set up to stop the test generation process for a fault when it becomes too costly. *In a practical sense, there is no difference between an undetectable fault f and a detectable one g that is not detected by an applied test set.* Clearly, g could be present in the circuit and hence invalidate the single-fault assumption.

Identifying redundancy is closely related to the problem of test generation. To show that a line is redundant means to prove that no test exists for the corresponding fault. The test generation problem belongs to a class of computationally difficult problems, referred to as *NP-complete* [Ibarra and Sahni 1975]. The traveling salesman problem is one famous member of this class [Horowitz and Sahni 1978]. Let n be the "size" of the problem. For the traveling salesman problem n is the number of cities to be visited; for test generation n is the number of gates in a circuit. An important question is whether there exists an algorithm that can solve any instance of a problem of size n using a number of operations proportional to n^r, where r is a finite constant. At present, no such polynomial-time algorithm is known for any *NP*-complete problem. These problems are related in the sense that either all of them or none of them can be solved by polynomial-time algorithms.

Although test generation (and identification of redundancy) is a computationally difficult problem, practical test generation algorithms usually run in polynomial time. The fact that the test generation problem is *NP*-complete means that polynomial time cannot be achieved in all instances, that is, any test generation algorithm may encounter a circuit with a fault that cannot be solved in polynomial time. Experience has shown that redundant faults are usually the ones that cause test generation algorithms to exhibit their worst-case behavior.

4.2.2 Sequential Circuits

Testing sequential circuits is considerably more difficult than testing combinational circuits. To detect a fault a test sequence is usually required, rather than a single input vector, and the response of a sequential circuit is a function of its initial state.

We will first illustrate a general form of fault detection for a sequential circuit. Let T be a test sequence and $R(q,T)$ be the response to T of a sequential circuit N starting in the initial state q. Now consider the circuit N_f obtained in the presence of the fault f. Similarly we denote by $R_f(q_f,T)$ the response of N_f to T starting in the initial state q_f.

Definition 4.2: A test sequence T **strongly detects** the fault f iff the output sequences $R(q,T)$ and $R_f(q_f,T)$ are different for every possible pair of initial states q and q_f.

Example 4.5: Figure 4.9 shows a synchronous circuit, its state table, and the output sequences obtained in response to the input sequence $T = 10111$, from the fault-free circuit and the circuits with the single faults α (line a s-a-1) and β (line b s-a-0).

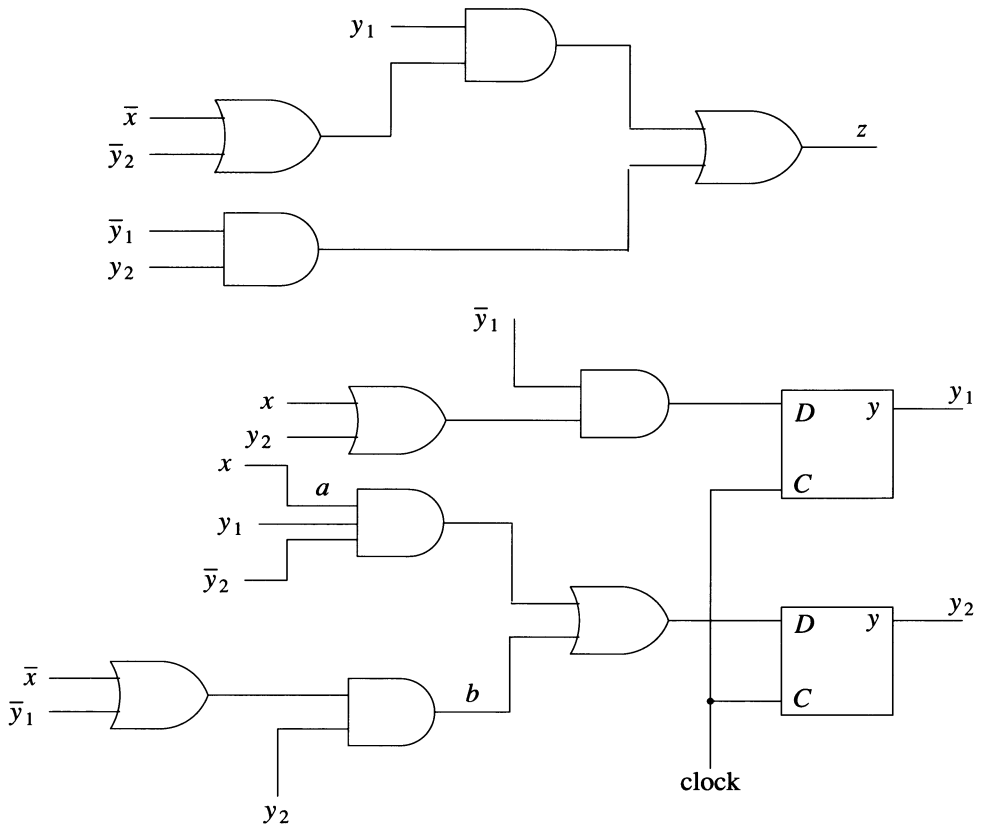
Since all sequences generated in the presence of β are different from those of the fault-free circuit, T strongly detects β. Since the sequences of the fault-free circuit in initial state B and the circuit with fault α in initial state B are identical, T does not strongly detect α. □

In this example, although T strongly detects β, the error symptom cannot be simply specified. That is, we cannot say that at some point in the output sequence the normal circuit will have a 1 output and the faulty circuit a 0 output or vice versa. Instead, we must list all possible responses of the normal and faulty machines. This is obviously not practical, as a circuit with n memory elements may have 2^n possible initial states. Moreover, we have to consider how the tester operates. A tester performing edge-pin testing with full comparison of the output results compares the obtained and the expected output sequences on a vector-by-vector basis. Hence, the expected output response must be known in advance. Practically feasible decisions on fault location and practical test generation algorithms cannot use ambiguity of the type "detection in either vector i or j." Thus the response R_f must also be predictable. Therefore, we will use the following (less general) concept of detection.

Definition 4.3: A test sequence T **detects** the fault f iff, for every possible pair of initial states q and q_f, the output sequences $R(q,T)$ and $R_f(q_f,T)$ are different for some specified vector $t_i \in T$.

To determine the vector t_i when an error caused by f can be observed at the primary outputs, independent of the initial states q and q_f, the testing experiment is usually divided into two distinct phases. In the first phase we apply an *initialization sequence* T_I such that at the end of T_I both N and N_f are brought to known states q_I and q_{If}. The output responses are ignored during T_I since they are not predictable. In the second phase we apply a sequence T'. Now both the expected response $R(q_I,T')$ and the faulty response $R_f(q_{If},T')$ are predictable. Usually t_i is taken as the first vector of T' for which an error is observed.

This type of testing is based on the fundamental assumption that such an initialization sequence T_I exists. Almost any circuit used in practice has an initialization sequence, which is employed to start its operation from a known state. Circuits are usually designed so that they can be easily initialized. An often used technique is to provide a common reset (or preset) line to every F/F. Then a single vector is needed to initialize



	x = 0	x = 1	y_1	y_2
A	A,0	D,0	0	0
B	C,1	C,1	0	1
C	B,1	A,0	1	1
D	A,1	B,1	1	0

Initial state	Output sequence		
	Fault-free	α (a *s-a*-1)	β (b *s-a*-0)
A	01011	01010	01101
B	11100	11100	11101
C	00011	00010	01010
D	11001	10010	11010

Figure 4.9 Output sequences as a function of initial state and fault

the circuit. However, an initialization sequence for the fault-free circuit N may fail to initialize some faulty circuit N_f. Such a fault f is said to *prevent initialization.*

Example 4.6: Consider the D F/F of Figure 4.10 (this is a typical configuration for a 1-bit counter) and the fault R s-a-1. While the fault-free circuit can be simply initialized by R=0, there is no way the faulty circuit can be initialized to a known state. □

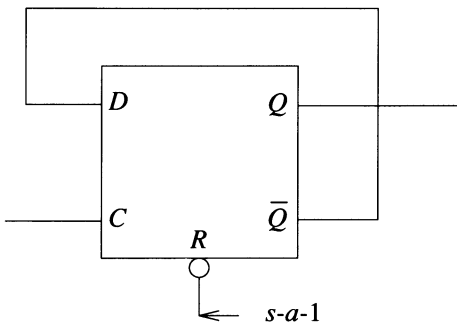


Figure 4.10 Example of a fault preventing initialization

One may derive a test sequence for a fault that prevents initialization by separately analyzing every possible initial state in the presence of that fault, but, in general, such an approach is impractical. Therefore faults that prevent initialization should be considered undetectable by edge-pin testing with full output comparison. However, unlike in a combinational circuit, this does not mean that the circuit is redundant [Abramovici and Breuer 1979]. Also it should be pointed out that a fault that prevents initialization may be detected by other types of testing, such as compact testing (to be discussed in a separate chapter).

4.3 Fault Equivalence and Fault Location

4.3.1 Combinational Circuits

Definition 4.4: Two faults f and g are said to be **functionally equivalent** iff $Z_f(x) = Z_g(x)$.

A test t is said to **distinguish** between two faults f and g if $Z_f(t) \neq Z_g(t)$; such faults are *distinguishable*. There is no test that can distinguish between two functionally equivalent faults. The relation of functional equivalence partitions the set of all possible faults into *functional equivalence classes*. For fault analysis it is sufficient to consider only one representative fault from every equivalence class.

For a single-output circuit, a test t that distinguishes between f and g makes $Z_f(t) = 0$ and $Z_g(t) = 1$ or vice versa. Thus, the set of all tests that distinguish between f and g is given by the solutions of the equation

$$Z_f(x) \oplus Z_g(x) = 1$$

Note that the faults f and g in Definition 4.4 are not restricted to the same fault universe. For example, in the circuit in Figure 4.11 the AND bridging fault between x and y is functionally equivalent to the stuck fault z s-a-0. Also an AND bridging fault between two complementary signals q and $\overline{q}$ is functionally equivalent to the multiple stuck fault $\{q\ s\text{-}a\text{-}0,\ \overline{q}\ s\text{-}a\text{-}0\}$. In general, however, we will limit the analysis of equivalence relations to those among faults of the same type.

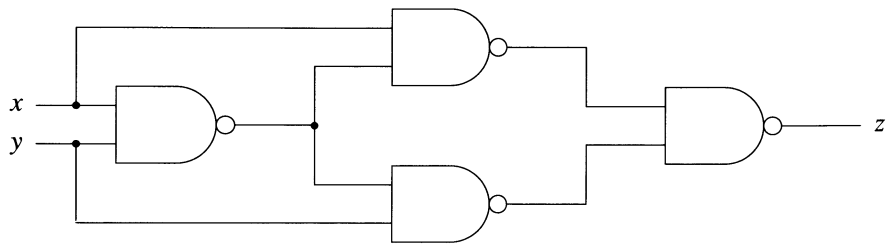


Figure 4.11

With any n-input gate we can associate $2(n+1)$ single stuck faults. For a NAND gate all the input s-a-0 faults and the output s-a-1 are functionally equivalent. In general, for a gate with controlling value c and inversion i, all the input s-a-c faults and the output s-a-$(c \oplus i)$ are functionally equivalent. Thus for an n-input gate ($n>1$) we need to consider only $n+2$ single stuck faults. This type of reduction of the set of faults to be analyzed based on equivalence relations is called **equivalence fault collapsing**. An example is shown in Figure 4.12(a) and (b), where a black (white) dot denotes a s-a-1 (s-a-0) fault.

If in addition to fault detection, the goal of testing is *fault location* as well, we need to apply a test that not only detects the detectable faults but also distinguishes among them as much as possible. A *complete location test set* distinguishes between every pair of distinguishable faults in a circuit.

It is convenient to consider that a fault-free circuit contains an *empty fault* denoted by Φ. Then $Z_\Phi(x) = Z(x)$. This artifact helps in understanding the relation between fault location and fault detection. Namely, fault detection is just a particular case of fault location, since a test that detects a fault f distinguishes between f and Φ. Undetectable faults are in the same equivalence class with Φ. Hence, *a complete location test set must include a complete detection test set.*

The presence of an undetectable fault may invalidate a complete location test set. If f and g are two distinguishable faults, they may become functionally equivalent in the presence of an undetectable fault (see Problem 4.8).

A complete location test set can diagnose a fault to within a functional equivalence class. This represents the maximal diagnostic resolution that can be achieved by edge-pin testing. In practice, large circuits are tested with test sets that are not

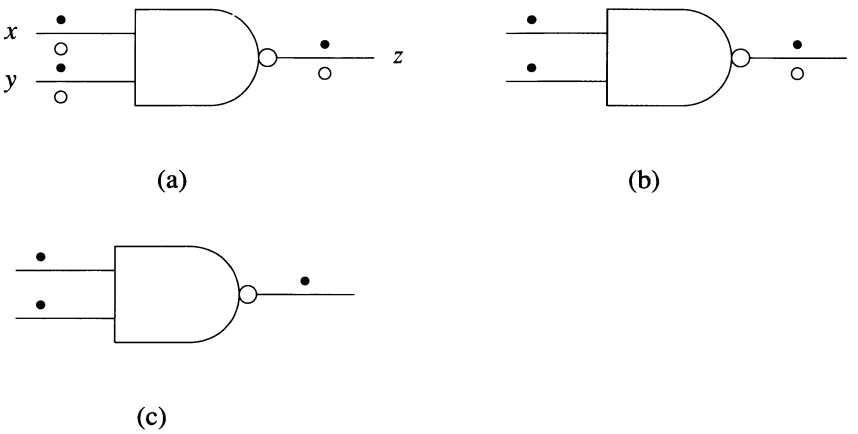


Figure 4.12 (a) NAND gate with uncollapsed fault set (b) Equivalence fault collapsing (c) Dominance fault collapsing

complete. Another equivalence relation can be used to characterize the resolution achieved by an arbitrary test set.

Definition 4.5: Two faults f and g are **functionally equivalent under a test set** T iff $Z_f(t) = Z_g(t)$ for every test $t \in T$.

Functional equivalence implies equivalence under any test set, but equivalence under a given test set does not imply functional equivalence (see Problem 4.9).

4.3.2 Sequential Circuits

Definition 4.6: Two faults f and g are said to be **strongly functionally equivalent** iff the corresponding sequential circuits N_f and N_g have equivalent state tables. (For state table equivalence refer to [Friedman and Menon 1975].)

Like the concept of strong detection, strong equivalence cannot be used in practice. The practical definition of functional equivalence is based on the responses of N_f and N_g to a test sequence T. We assume the same type of testing experiment as discussed for detection; namely, T contains first an initialization sequence T_I that brings N_f and N_g to known states q_{If} and q_{Ig}. The output responses are not monitored during the application of T_I since they are not predictable. Let T' be the sequence applied after T_I.

Definition 4.7: Two faults f and g are said to be **functionally equivalent** iff $R_f(q_{If},T') = R_g(q_{Ig},T')$ for any T'.

Again, this definition does not apply to faults preventing initialization.

Similarly, functional equivalence under a test sequence $T = \{T_I,T'\}$ means that $R_f(q_{If},T') = R_g(q_{Ig},T')$.

4.4 Fault Dominance

4.4.1 Combinational Circuits

If the objective of a testing experiment is limited to fault detection only, then, in addition to fault equivalence, another fault relation can be used to reduce the number of faults that must be considered.

Definition 4.8: Let T_g be the set of all tests that detect a fault g. We say that a fault f **dominates** the fault g iff f and g are functionally equivalent under T_g.

In other words, if f dominates g, then any test t that detects g, i.e., $Z_g(t) \neq Z(t)$, will also detect f (on the same primary outputs) because $Z_f(t) = Z_g(t)$. Therefore, for fault detection it is unnecessary to consider the dominating fault f, since by deriving a test to detect g we automatically obtain a test that detects f as well (see Figure 4.13).

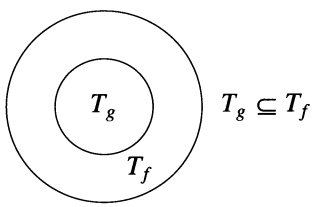


Figure 4.13 The sets T_f and T_g when f dominates g

For example, consider the NAND gate of Figure 4.12 and let g be y s-a-1 and f be z s-a-0. The set T_g consists of only one test, namely 10, and this also detects f. Then z s-a-0 dominates y s-a-1. In general, for a gate with controlling value c and inversion i, the output $s\text{-}a\text{-}(\overline{c} \oplus i)$ fault dominates any input $s\text{-}a\text{-}\overline{c}$ fault. Then the output fault can be removed from the set of faults we consider for test generation (see Figure 4.12(c)). This type of reduction of the set of faults to be analyzed based on dominance relations is called **dominance fault collapsing**.

It is interesting to observe that we can have two faults, f and g, such that any test that detects g also detects f (i.e., $T_g \subseteq T_f$), without f dominating g. Consider, for example, the circuit of Figure 4.14. Let f be z_2 s-a-0 and g be y_1 s-a-1. The set T_g consists only of the test 10, which also detects f (on a different primary output). But according to Definition 4.8, f does not dominate g. Although for fault detection it is not necessary to consider f, it would be difficult to determine this fact from an analysis of the circuit.

When choosing a fault model it is important to select one whose faults are generally dominated by faults of other fault models, because a test set detecting the faults of the chosen model will also detect many other faults that are not even explicitly considered. The best fault model with such a property appears to be the single stuck-fault model.

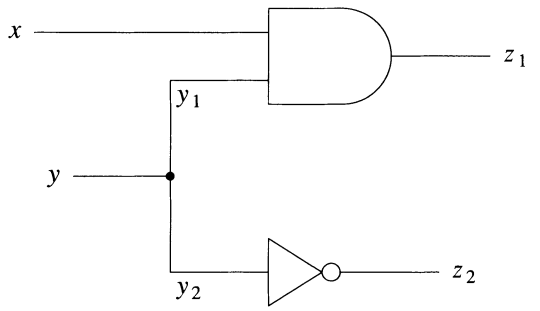


Figure 4.14

4.4.2 Sequential Circuits

Although the concept of dominance can be extended to sequential circuits, its applicability in practice is difficult, since dominance relations in a combinational circuit N may not remain valid when N is embedded in a sequential circuit. The following example illustrates this phenomenon.

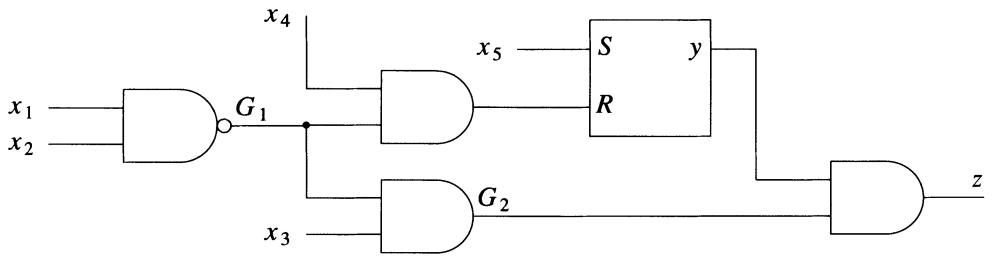
Example 4.7: Consider the circuit of Figure 4.15. Assume that initially y=1 and consider the test sequence shown in the figure. For the fault-free circuit the output sequence generated is 0000. For the fault α (x_2 *s-a-*1) the first input fails to reset y and the fourth input propagates this error to z. Thus the generated output sequence is 0001 and x_2 *s-a-*1 is detected. Now consider the same test sequence and the fault β (G_1 *s-a-*0), which dominates α in a combinational circuit sense. The first input again fails to reset y. However, the fourth input generates an erroneous 0 at G_2 and the two effects of this single fault — the one stored in the F/F and the one propagating along the path (G_1, G_2) — cancel each other, and the fault is not detected. □

While equivalence fault-collapsing techniques for combinational circuits remain valid for sequential circuits, dominance fault-collapsing techniques are no longer applicable.

4.5 The Single Stuck-Fault Model

The single-stuck fault (SSF) model is also referred to as the *classical* or *standard* fault model because it has been the first and the most widely studied and used. Although its validity is not universal, its usefulness results from the following attributes:

- It represents many different physical faults (see, for example, [Timoc *et al.* 1983]).
- It is independent of technology, as the concept of a signal line being stuck at a logic value can be applied to any structural model.
- Experience has shown that tests that detect SSFs detect many nonclassical faults as well.



	x_1	x_2	x_3	x_4	x_5
t_1	1	0	0	1	0
t_2	1	0	0	0	0
t_3	0	1	0	0	0
t_4	0	1	1	0	0

Figure 4.15

- Compared to other fault models, the number of SSFs in a circuit is small. Moreover, the number of faults to be explicitly analyzed can be reduced by fault-collapsing techniques.
- SSFs can be used to model other type of faults.

The last point is illustrated in Figure 4.16. To model a fault that changes the behavior of the signal line z, we add to the original circuit a multiplexor (selector) that realizes the function

$$z' = z \quad \text{if } f = 0$$
$$z' = z_f \quad \text{if } f = 1$$

The new circuit operates identically to the original circuit for f=0 and can realize any faulty function z_f by inserting the fault f *s-a*-1. For example, connecting x to z_f would create the effect of a functional fault that changes the function of the inverter from $z=\bar{x}$ to $z=x$. Connecting x to z_f via an inverter with a different delay would create the effect of a delay fault. Although flexible, this approach to nonclassical fault modeling is limited by the need to increase significantly the size of the model.

When considering an explicit fault model it is important to know how large is the fault universe it defines. If there are n lines on which SSFs can be defined, the number of possible faults is $2n$. To determine n we have to consider every fanout branch as a separate line. We will do this computation for a gate-level model. Every signal source i (gate output or primary input) with a fanout count of f_i contributes k_i lines to the number of possible fault sites, where k_i is defined as follows:

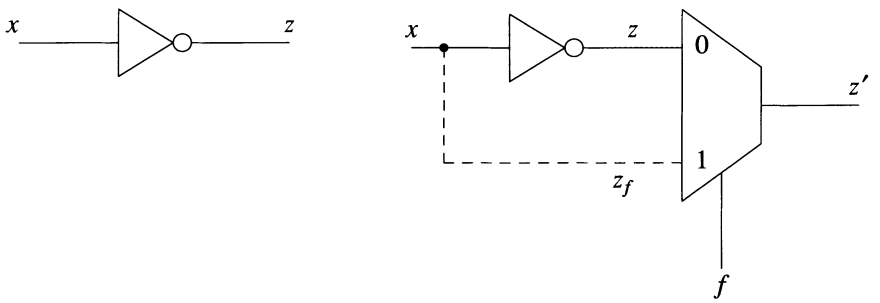


Figure 4.16 Model modification for nonclassical fault modeling

$$k_i = \begin{cases} 1 & \text{if } f_i = 1 \\ 1+f_i & \text{if } f_i > 1 \end{cases} \tag{4.2}$$

The number of possible fault sites is

$$n = \sum_i k_i \tag{4.3}$$

where the summation is over all the signal sources in the circuit. Let us define a variable q_i by

$$q_i = \begin{cases} 1 & \text{if } f_i = 1 \\ 0 & \text{if } f_i > 1 \end{cases} \tag{4.4}$$

Then (4.3) becomes

$$n = \sum_i (1 + f_i - q_i) \tag{4.5}$$

Let us denote the number of gates by G and the number of primary inputs by I. The average fanout count in the circuit is given by

$$f = \frac{\sum_i f_i}{G + I} \tag{4.6}$$

The fraction of signal sources with only one fanout can be expressed as

$$q = \frac{\sum_i q_i}{G + I} \tag{4.7}$$

With this notation we obtain from (4.5)

$$n = (G + I)\,(1 + f - q) \tag{4.8}$$

(Just as a check, apply (4.8) to a fanout-free circuit). In a large circuit, usually $G>>I$ and $q>0.5$, so the dominating factor in (4.8) is Gf. Thus we can conclude that the number of SSFs is slightly larger than $2Gf$. It is important to observe that it *depends on both the gate count and on the average fanout count.*

In the previous section we noted that the number of faults to be analyzed can be reduced by fault collapsing based on equivalence and dominance relations. Functional equivalence relations, however, cannot be directly applied for this purpose, because determining whether two arbitrary faults are functionally equivalent is an *NP*-complete problem [Goundan 1978]. For example, there is no simple way to determine that the faults *c s-a-*1 and *d s-a-*1 in Figure 4.17 are functionally equivalent. (We can do this by computing the two faulty functions and showing that they are identical.)

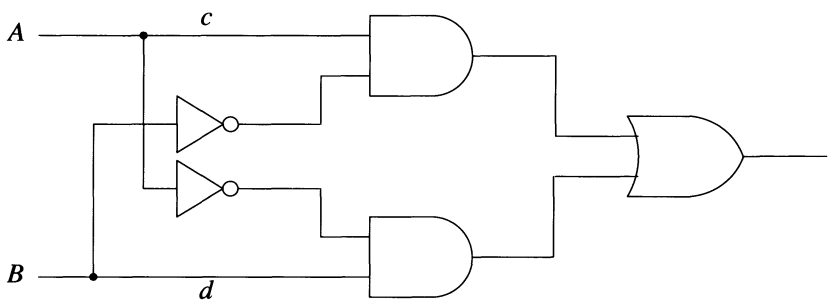


Figure 4.17

What we can do is to determine equivalent faults that are structurally related. The relation on which this analysis relies is called **structural equivalence**, and it is defined as follows. In the circuit N_f, the presence of the stuck fault f creates a set of lines with constant values. By removing all these lines (except primary outputs) we obtain a simplified circuit $S(N_f)$, which realizes the same function as N_f. Two faults f and g are said to be structurally equivalent if the corresponding simplified circuits $S(N_f)$ and $S(N_g)$ are identical. An example is shown in Figure 4.18. Obviously, structurally equivalent faults are also functionally equivalent. But, as illustrated by *c s-a-*1 and *d s-a-*1 in Figure 4.17, there exist functionally equivalent faults that are not structurally equivalent. The existence of such faults is related to the presence of reconvergent fanout [McCluskey and Clegg 1971] (see Problem 4.13).

The advantage of structural equivalence relations is that they allow fault collapsing to be done as a simple *local analysis* based on the structure of the circuit, while functional equivalence relations imply a global analysis based on the function of the circuit. Figure 4.19 illustrates the process of fault collapsing based on structural equivalence relations. Conceptually, we start by inserting *s-a-*1 and *s-a-*0 faults on every signal source (gate output or primary input) and destination (gate input). This is shown in Figure 4.19(a), where a black (white) dot denotes a *s-a-*1 (*s-a-*0) fault. Then we traverse the circuit and construct structural equivalence classes along the way. For a signal line with a fanout count of 1, the faults inserted at its source are structurally equivalent to the corresponding faults at its destination. For a gate with controlling value c and inversion i, any *s-a-c* input fault is structurally equivalent to the output *s-a-*($c \oplus i$). A *s-a-*0 (*s-a-*1) input fault of an inverter is structurally equivalent to the

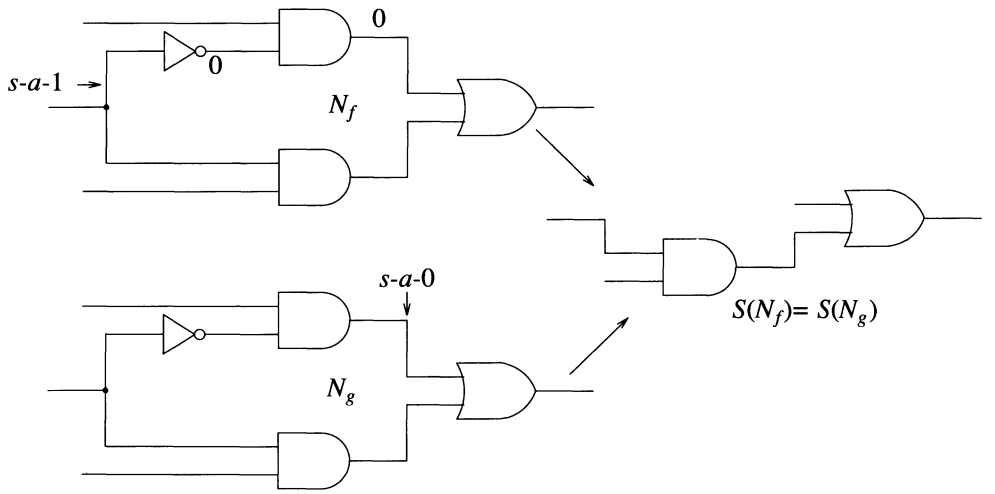


Figure 4.18 Illustration for structural fault equivalence

output *s-a-1* (*s-a-0*). Finally, from every equivalence class we retain one fault as representative (Figure 4.19(b)).

A structural equivalence class determined in this way is confined to a fanout-free region of the circuit. The reason is that in general, a stem *s-a-v* fault is not functionally equivalent with a *s-a-v* fault on any of its fanout branches. Reconvergent fanout, however, may create structurally equivalent faults in different fanout-free regions; Figure 4.20 illustrates such a case. Thus our method of equivalence fault collapsing will not identify faults such as *b s-a-0* and *f s-a-0* as structurally equivalent. In such a case, the obtained structural equivalence classes are not maximal; i.e., there are at least two classes that can be further merged. However, the potential gain achievable by extending the fault collapsing across fanout-free regions is small and does not justify the cost of the additional analysis required.

Although the equivalence classes obtained by structural equivalence fault collapsing are not maximal, this process nevertheless achieves a substantial reduction of the initial set of $2n$ faults, where n is given by (4.8). For every gate j with g_j inputs, it eliminates m_j input faults, where

$$m_j = \begin{cases} g_j & \text{if } g_j > 1 \\ 2 & \text{if } g_j = 1 \end{cases} \tag{4.9}$$

The number of faults eliminated is

$$m = \sum_j m_j \tag{4.10}$$

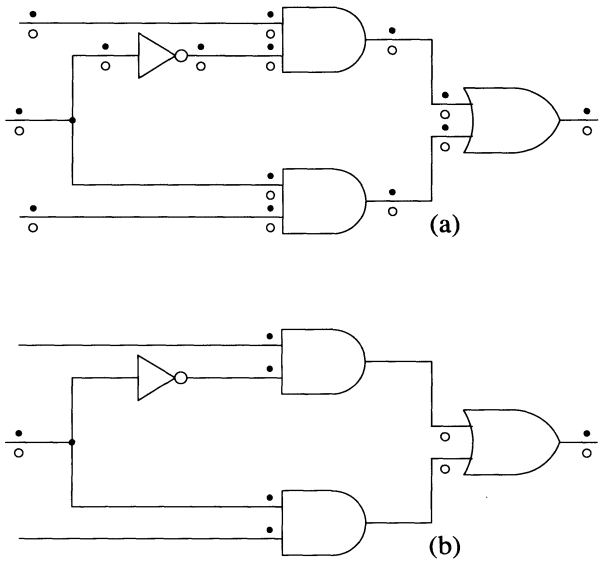


Figure 4.19 Example of structural equivalence fault collapsing

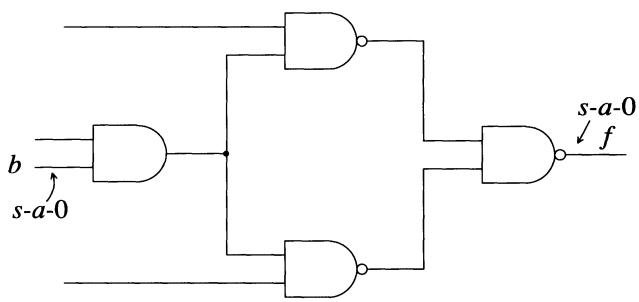


Figure 4.20

where the summation is over all the gates in the circuit. Let us define a variable p_j by

$$p_j = \begin{cases} 0 & \text{if } g_j > 1 \\ 1 & \text{if } g_j = 1 \end{cases} \tag{4.11}$$

Then (4.10) becomes

$$m = \sum_j (g_j + p_j) \tag{4.12}$$

The average input count in the circuit is given by

$$g = \frac{\sum_j g_j}{G} \tag{4.13}$$

The fraction of the gates with only one input can be expressed as

$$p = \frac{\sum_j p_j}{G} \tag{4.14}$$

With this notation we obtain from (4.12)

$$m = G(g+p) \tag{4.15}$$

Since the average input count g is nearly equal to the average fanout count f, it follows that more than Gf faults are eliminated. Thus structural equivalence fault collapsing reduces the initial set of faults by about 50 percent.

If we are interested only in fault detection, we can also do dominance fault collapsing. This process is based on the following two theorems.

Theorem 4.1: In a fanout-free combinational circuit C, any test set that detects all SSFs on the primary inputs of C detects all SSFs in C.

Proof: Assume, by contradiction, that a test set T detects all SSFs on the primary inputs of C but does not detect all internal SSFs. Because every line in C has only one fanout, it is sufficient to consider internal faults only on gate outputs. Since T detects all SSFs on the primary inputs, there must be some gate G in C such that T detects all faults in the circuit feeding G but does not detect some output fault f. First assume G is an AND gate. Then f cannot be the s-a-0 fault, since this is equivalent to any input s-a-0. Also f cannot be the s-a-1 fault, since this dominates any input s-a-1. Hence such a fault f does not exist. A similar argument holds if G is an OR, NAND, or NOR gate. Therefore all the internal faults in C are detected. □

A similar proof mechanism can be used for the following theorem.

Theorem 4.2: In a combinational circuit C any test set that detects all SSFs on the primary inputs and the fanout branches of C detects all SSFs in C. □

The primary inputs and the fanout branches are called *checkpoints*. The number of checkpoints is

$$r = I + \sum_i (f_i - q_i) = I + (G+I)(f-q) \tag{4.16}$$

Thus dominance fault collapsing reduces the number of faults dealt with from $2n$ to $2r$. Usually the number of fanout branches — $(G+I)(f-q)$ — is much larger than the number I of primary inputs, so we can approximate

$$r \simeq (G+I)(f-q) \tag{4.17}$$

The fraction of faults eliminated is

$$1 - \frac{2r}{2n} = 1 - \frac{f-q}{1+f-q} = \frac{1}{1+f-q} \tag{4.18}$$

For example, for typical values such as $f = 2.5$ and $q = 0.7$, we eliminate about 40 percent of the faults. The set of checkpoint faults can be further collapsed by using structural equivalence and dominance relations, as illustrated in the following example.

Example 4.8: The circuit of Figure 4.21 has 24 SSFs and 14 checkpoint faults (10 on the primary inputs — *a*, *b*, *c*, *d*, *e* — and 4 on the fanout branches *g* and *h*). Since *a* *s-a-0* and *b* *s-a-0* are equivalent, we can delete the latter. Similarly, we can delete *d* *s-a-0*, which is equivalent to *h* *s-a-0*. The fault *g* *s-a-1* is equivalent to *f* *s-a-1*, which dominates *a* *s-a-1*. Therefore, *g* *s-a-1* can be eliminated. Similarly, *e* *s-a-1* is equivalent to *i* *s-a-1*, which dominates *h* *s-a-1*; hence *e* *s-a-1* can be eliminated. The original set of 24 faults has thus been reduced to 10. □

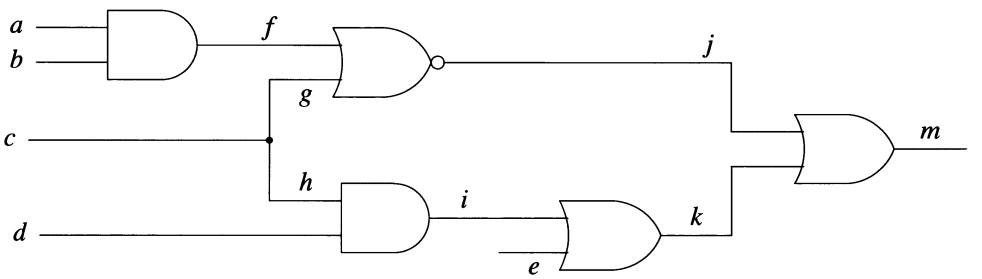


Figure 4.21

Based on Theorem 4.2, many test generation systems generate tests explicitly only for the checkpoint faults of a circuit. But Theorem 4.2 is meaningful only for an irredundant circuit. In a redundant circuit, some of the checkpoint faults are undetectable. If we consider only checkpoint faults and we generate a test set that detects all detectable checkpoint faults, this test set is not guaranteed to detect all detectable SSFs of the circuit; in such a case, additional tests may be needed to obtain a complete detection test set [Abramovici *et al.* 1986].

Understanding the relations between faults on a stem line and on its fanout branches is important in several problems that we will discuss later. Clearly a stem *s-a-v* is equivalent to the multiple fault composed of all its fanout branches *s-a-v*. But, in general, neither equivalence nor dominance relations exist between a stem *s-a-v* and an individual fanout branch *s-a-v*. Figure 4.22 shows an example [Hayes 1979] in which a stem fault (*j* *s-a-0*) is detected, but the (single) faults on its fanout branches (*k* *s-a-0* and *m* *s-a-0*) are not. Figure 4.23 illustrates the opposite situation [Abramovici *et al.* 1984], in which faults on all fanout branches (x_1 *s-a-0* and x_2 *s-a-0*) are detected, but the stem fault (*x* *s-a-0*) is not.

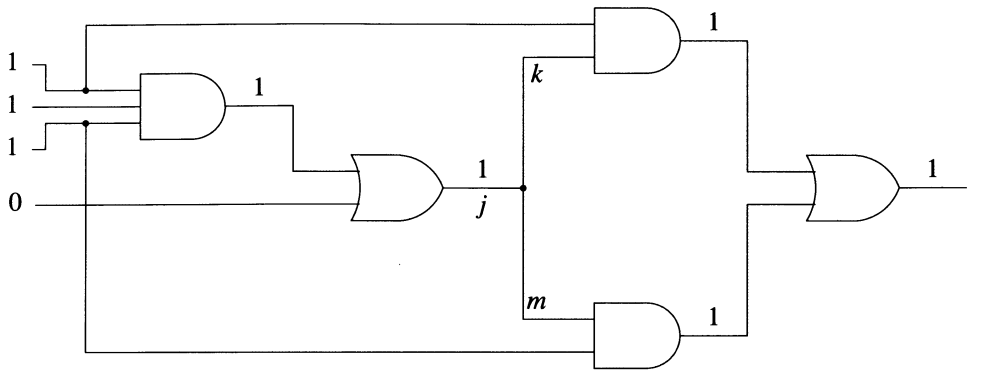


Figure 4.22

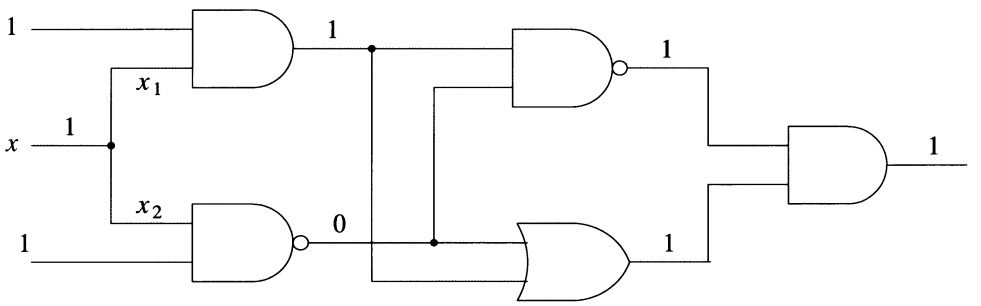


Figure 4.23

4.6 The Multiple Stuck-Fault Model

The multiple stuck-fault (MSF) model is a straightforward extension of the SSF model in which several lines can be simultaneously stuck. If we denote by n the number of possible SSF sites (given by formula (4.8)), there are $2n$ possible SSFs, but there are 3^n-1 possible MSFs (which include the SSFs). This figure assumes that any MSF can occur, including the one consisting of all lines simultaneously stuck. If we assume that the multiplicity of a fault, i.e., the number of lines simultaneously stuck, is no greater than a constant k, then the number of possible MSFs is $\sum_{i=1}^{k} \binom{n}{i} 2^i$. This is usually too large a number to allow us to deal explicitly with all multiple faults. For example, the

number of double faults (k=2) in a circuit with n=1000 possible fault sites is about half a million.

Let us first consider the question of why we need to consider MSFs altogether. Since a multiple fault F is just a set $\{f_1,f_2,...,f_k\}$ of single faults f_i, why isn't F detected by the tests that detect the single faults f_i? The explanation is provided by the *masking relations* among faults.

Definition 4.9: Let T_g be the set of all tests that detect a fault g. We say that a fault f **functionally masks** the fault g iff the multiple fault $\{f,g\}$ is not detected by any test in T_g.

Example 4.9: In the circuit of Figure 4.24 the test 011 is the only test that detects the fault c s-a-0. The same test does not detect the multiple fault $\{c$ s-a-0, a s-a-1$\}$. Thus a s-a-1 masks c s-a-0. □

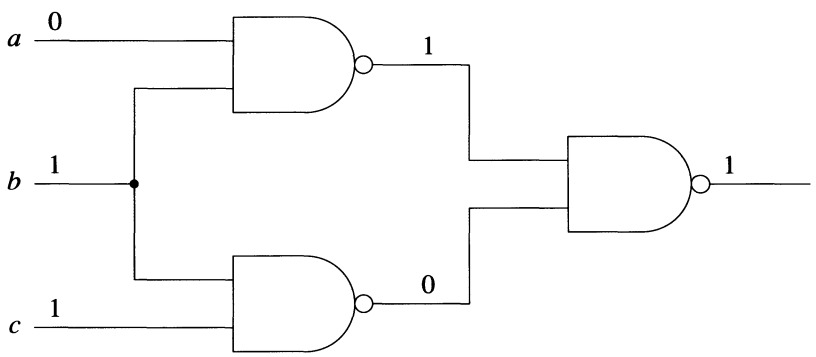


Figure 4.24

Masking can also be defined with respect to a test set T.

Definition 4.10: Let $T_g' \subseteq T$ be the set of all tests in T that detect a fault g. We say that a fault f **masks** the fault g **under a test set** T iff the multiple fault $\{f,g\}$ is not detected by any test in T_g'.

Functional masking implies masking under any test set, but the converse statement is not always true.

Masking relations can also be defined among different type of faults. In Example 4.4 we have an undetectable bridging fault that masks a detectable SSF under a complete test set for SSFs.

If f masks g, then the fault $\{f,g\}$ is not detected by the tests that detect g alone. But $\{f,g\}$ may be detected by other tests. This is the case in Example 4.9, where the fault $\{c$ s-a-0, a s-a-1$\}$ is detected by the test 010.

An important problem is, given a complete test set T for single faults, can there exist a multiple fault $F = \{f_1, f_2, ..., f_k\}$ such that F is not detected by T? (Remember that T detects every f_i alone.) The answer is provided by the following example.

Example 4.10: The test set T = {1111, 0111, 1110, 1001, 1010, 0101} detects every SSF in the circuit of Figure 4.25. Let f be B s-a-1 and g be C s-a-1. The only test in T that detects the single faults f and g is 1001. However, the multiple fault $\{f,g\}$ is not detected because under the test 1001, f masks g and g masks f. □

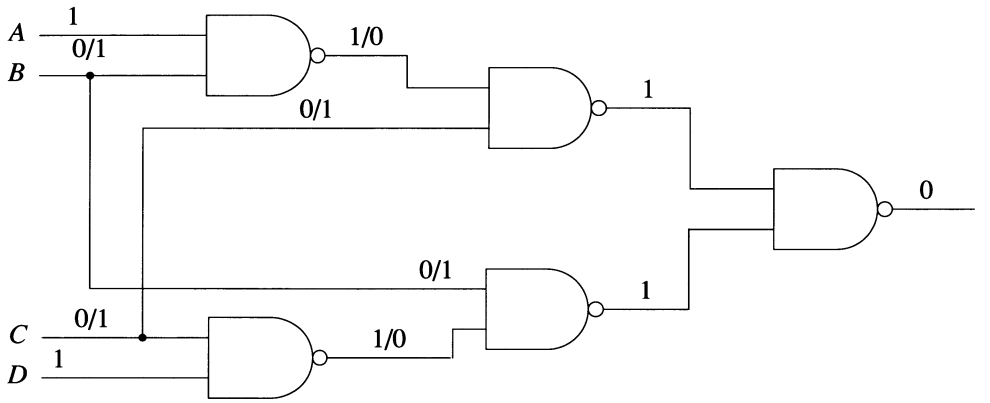


Figure 4.25

In the above example, a multiple fault F is not detected by a complete test set T for single faults because of *circular masking relations* under T among the components of F. Circular functional masking relations may result in an undetectable multiple fault F, as shown in the following example [Smith 1978].

Example 4.11: In the circuit of Figure 4.26, all SSFs are detectable. Let f be D s-a-1 and g be E s-a-1. One can verify that f functionally masks g and vice versa and that the multiple fault $\{f,g\}$ is undetectable. (This represents another type of redundancy, called *multiple-line redundancy*.) □

Note that the existence of circular masking relations among the SSF components f_i of a MSF F is a necessary but not a sufficient condition for F to be undetectable [Smith 1979].

An important practical question is: What percentage of the MSFs can escape detection by a test set designed to detect SSFs? The answer depends on the structure (but not on the size) of the circuit. Namely, the following results have been obtained for combinational circuits:

1. In an irredundant two-level circuit, *any* complete test set for SSFs also detects all MSFs [Kohavi and Kohavi 1972].

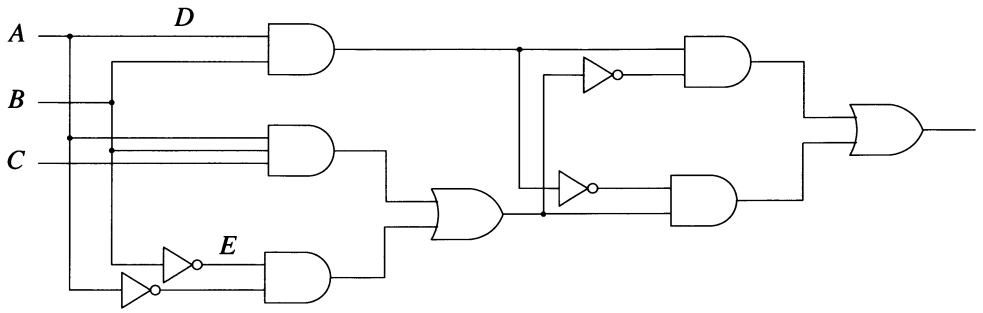


Figure 4.26

2. In a fanout-free circuit, any complete test set for SSFs detects all double and triple faults, and there exists a complete test set for SSFs that detects all MSFs [Hayes 1971].
3. In an internal fanout-free circuit (i.e., a circuit in which only the primary inputs may be stems), any complete test set for SSFs detects at least 98 percent of all MSFs of multiplicity less than 6 [Agarwal and Fung 1981].
4. In an internal fanout-free circuit C, any complete test set for SSFs detects all MSFs unless C contains a subcircuit with the same interconnection pattern as the 5-gate circuit shown in Figure 4.25 [Schertz and Metze 1972].
5. A test set that detects all MSFs defined on all primary inputs without fanout and all fanout branches of a circuit C detects all multiple faults in C [Bossen and Hong 1971].

Since a test set T designed for detecting SSFs also detects most of the MSFs, for MSF analysis we would like to deal directly only with those MSFs *not* detected by T. Algorithms to identify MSFs not detected by a test set are presented in [Cha 1979, Abramovici and Breuer 1980] for combinational circuits and in [Abramovici and Breuer 1982] for sequential circuits.

Unless circuits are used in extremely high-reliability applications, the MSF model is mainly of theoretical interest. Although the MSF model is generally more realistic than the SSF model, the use of the latter is justified since circular masking conditions are seldom encountered in practice.

Jacob and Biswas [1987] identified SSFs that are *guaranteed to be detected* (GTBD), that is, they can be detected regardless of the presence of other faults in the circuit. For example, any fault on a primary output is GTBD. All the MSFs containing a SSF f that is GTBD will also be detected by the tests for f. It can be shown that a very large fraction of the MSFs (at least 99.6 percent for circuits with three or more primary outputs) contain GTBD faults, which means that circular masking cannot occur. This result takes into account MSFs of any multiplicity, and it reflects the contribution of the large number of MSFs of higher multiplicity, which are more likely to contain

GTBD faults. In general, MSFs of lower multiplicity are more difficult to detect. However, experimental results obtained by Hughes and McCluskey [1984], simulating the 74LS181 4-bit ALU, show that complete detection test sets for SSFs detect all (or more than 99.9 percent) of the double faults and most of the sampled triple and quadruple faults.

4.7 Stuck RTL Variables

A straightforward extension of the stuck-fault model is to replace the concept of a signal line that is stuck with that of an internal RTL variable being stuck. (Stuck I/O variables of functional models are covered by the pin stuck-fault model.) We can further differentiate between *data faults* and *control faults*, depending on the type of the stuck variable. Typical data faults are register or memory bits that are stuck. Control faults are defined on variables that control conditional operations. Thus in the statement

$$\textbf{if } (X \textbf{ and } CLK) \textbf{ then } A = B$$

if X is s-a-1 the transfer $A = B$ will always occur when $CLK = 1$, and it will never occur if X is s-a-0. More generally, we can allow the result of any expression that is part of a condition (or the entire condition itself) to have a stuck fault. In the above example, if the expression (X and CLK) is s-a-1, the transfer $A=B$ becomes unconditional. This functional fault model is often used because it is easy to integrate with gate-level stuck-fault modeling.

4.8 Fault Variables

The effect of a fault f on the behavior of a system can be explicitly described using RTL constructs. To do this we introduce a *fault variable* f to reflect the existence of the fault f [Menon and Chappel 1978]. Then we can use f as a condition to switch-in the behavior in the presence of f, as illustrated in

$$\begin{aligned}&\textbf{if } f \textbf{ then } A = B-C\\ &\quad\textbf{else } A = B+C\end{aligned}$$

where $A = B+C$ and $A = B-C$ represent, respectively, the fault-free and the faulty operations. The fault variable f is treated as a state variable whose value is always 0 for normal operation. Thus the fault f can be modeled as a s-a-1 fault on the internal variable f. This is similar to the technique illustrated in Figure 4.16.

This general method can explicitly model any functional fault. The next example describes a (pattern-sensitive) fault f whose effect — complementing Z — occurs only in the presence of a certain pattern ($DREG$=1111):

$$\textbf{if } (f \textbf{ and } DREG{=}1111) \textbf{ then } Z = \overline{Z}$$

We can also apply this technique to model delay faults, as illustrated in

$$\begin{aligned}&\textbf{if } f \textbf{ then } C = A+B \textbf{ delay } 40\\ &\quad\textbf{else } C = A+B \textbf{ delay } 10\end{aligned}$$

Functional fault modeling based on fault variables is limited by the need to define explicitly the faulty behavior, and it is applicable only to a small number of specific faults of interest.

REFERENCES

[Abraham and Fuchs 1986] J. A. Abraham and W. K. Fuchs, "Fault and Error Models for VLSI," *Proc. of the IEEE*, Vol. 75, No. 5, pp. 639-654, May, 1986.

[Abramovici *et al.* 1984] M. Abramovici, P. R. Menon, and D. T. Miller, "Critical Path Tracing: An Alternative to Fault Simulation," *IEEE Design & Test of Computers,* Vol. 1, No. 1, pp. 83-93, February, 1984.

[Abramovici *et al.* 1986] M. Abramovici, P. R. Menon, and D. T. Miller, "Checkpoint Faults Are Not Sufficient Target Faults for Test Generation," *IEEE Trans. on Computers*, Vol. C-35, No. 8, pp. 769-771, August, 1986.

[Abramovici and Breuer 1979] M. Abramovici and M. A. Breuer, "On Redundancy and Fault Detection in Sequential Circuits," *IEEE Trans. on Computers*, Vol. C-28, No. 11, pp. 864-865, November, 1979.

[Abramovici and Breuer 1980] M. Abramovici and M. A. Breuer, "Multiple Fault Diagnosis in Combinational Circuits Based on an Effect-Cause Analysis," *IEEE Trans. on Computers*, Vol. C-29, No. 6, pp. 451-460, June, 1980.

[Abramovici and Breuer 1982] M. Abramovici and M. A. Breuer, "Fault Diagnosis in Synchronous Sequential Circuits Based on an Effect-Cause Analysis," *IEEE Trans. on Computers*, Vol. C-31, No. 12, pp. 1165-1172, December, 1982.

[Agarwal and Fung 1981] V. K. Agarwal and A. S. F. Fung, "Multiple Fault Testing of Large Circuits by Single Fault Test Sets," *IEEE Trans. on Computers*, Vol. C-30, No. 11, pp. 855-865, November, 1981.

[Banerjee and Abraham 1984] P. Banerjee and J. A. Abraham, "Characterization and Testing of Physical Failures in MOS Logic Circuits," *IEEE Design & Test of Computers*, Vol. 1, No. 4, pp. 76-86, August, 1984.

[Bossen and Hong 1971] D. C. Bossen and S. J. Hong, "Cause-Effect Analysis for Multiple Fault Detection in Combination Networks," *IEEE Trans. on Computers*, Vol. C-20, No. 11, pp. 1252-1257, November, 1971.

[Breuer 1973] M. A. Breuer, "Testing for Intermittent Faults in Digital Circuits," *IEEE Trans. on Computers*, Vol. C-22, No. 3, pp. 241-246, March, 1973.

[Case 1976] G. R. Case, "Analysis of Actual Fault Mechanisms in CMOS Logic Gates," *Proc. 13th Design Automation Conf.*, pp. 265-270, June, 1976.

[Cha 1979] C. W. Cha, "Multiple Fault Diagnosis in Combinational Networks," *Proc. 16th Design Automation Conf.*, pp. 149-155, June, 1979.

[Ferguson and Shen 1988] F. J. Ferguson and J. P. Shen, "A CMOS Fault Extractor for Inductive Fault Analysis," *IEEE Trans. on Computer-Aided Design*, Vol. 7, No. 11, pp. 1181-1194, November, 1988.

[Friedman 1967] A. D. Friedman, "Fault Detection in Redundant Circuits," *IEEE Trans. on Electronic Computers*, Vol. EC-16, pp. 99-100, February, 1967.

[Friedman and Menon 1975] A. D. Friedman and P. R. Menon, *Theory and Design of Switching Circuits*, Computer Science Press, Woodland Hills, California, 1975.

[Galiay *et al.* 1980] J. Galiay, Y. Crouzet, and M. Vergniault, "Physical Versus Logical Fault Models in MOS LSI Circuits: Impact on Their Testability," *IEEE Trans. on Computers*, Vol. C-29, No. 6, pp. 527-531, June, 1980.

[Goundan 1978] A. Goundan, *Fault Equivalence in Logic Networks*, Ph.D. Thesis, University of Southern California, March, 1978.

[Hayes 1971] J. P. Hayes, "A NAND Model for Fault Diagnosis in Combinational Logic Networks," *IEEE Trans. on Computers*, Vol. C-20, No. 12, pp. 1496-1506, December, 1971.

[Hayes 1976] J. P. Hayes, "On the Properties of Irredundant Logic Networks," *IEEE Trans. on Computers*, Vol. C-25, No. 9, pp. 884-892, September, 1976.

[Hayes 1977] J. P. Hayes, "Modeling Faults in Digital Logic Circuits," in *Rational Fault Analysis*, R. Saeks and S. R. Liberty, eds., Marcel Dekker, New York, pp. 78-95, 1977.

[Hayes 1979] J. P. Hayes, "Test Generation Using Equivalent Normal Forms," *Journal of Design Automation and Fault-Tolerant Computing*, Vol. 3, No. 3-4, pp. 131-154, Winter, 1979.

[Horowitz and Sahni 1978] E. Horowitz and S. Sahni, *Fundamentals of Computer Algorithms*, Computer Science Press, Rockville, Maryland, 1978.

[Hughes and McCluskey 1984] J. L. A. Hughes and E. J. McCluskey, "An Analysis of the Multiple Fault Detection Capabilities of Single Stuck-at Fault Test Sets," *Proc. Intn'l. Test Conf.*, pp. 52-58, October, 1984.

[Ibarra and Sahni 1975] O. H. Ibarra and S. Sahni, "Polynomially Complete Fault Detection Problems," *IEEE Trans. on Computers*, Vol. C-24, No. 3, pp. 242-249, March, 1975.

[Jacob and Biswas 1987] J. Jacob and N. N. Biswas, "GTBD Faults and Lower Bounds on Multiple Fault Coverage of Single Fault Test Sets," *Proc. Intn'l. Test Conf.*, pp. 849-855, September, 1987.

[Kohavi and Kohavi 1972] I. Kohavi and Z. Kohavi, "Detection of Multiple Faults in Combinational Logic Networks," *IEEE Trans. on Computers*, Vol. C-21, No. 6, pp. 556-668, June, 1972.

[Kodandapani and Pradhan 1980] K. L. Kodandapani and D. K. Pradhan, "Undetectability of Bridging Faults and Validity of Stuck-At Fault Test Sets," *IEEE Trans. on Computers*, Vol. C-29, No. 1, pp. 55-59, January, 1980.

[McCluskey and Clegg 1971] E. J. McCluskey and F. W. Clegg, "Fault Equivalence in Combinational Logic Networks," *IEEE Trans. on Computers*, Vol. C-20, No. 11, pp. 1286-1293, November, 1971.

[Menon and Chappell 1978] P. R. Menon and S. G. Chappell, "Deductive Fault Simulation with Functional Blocks," *IEEE Trans. on Computers*, Vol. C-27, No. 8, pp. 689-695, August, 1978.

[Partridge 1980] J. Partridge, "Testing for Bipolar Integrated Circuit Failure Modes," *Digest of Papers 1980 Test Conf.*, pp. 397-406, November, 1980.

[Savir 1980] J. Savir, "Detection of Single Intermittent Faults in Sequential Circuits," *IEEE Trans. on Computers*, Vol. C-29, No. 7, pp. 673-678, July, 1980.

[Schertz and Metze 1972] D. R. Schertz and G. Metze, "A New Representation of Faults in Combinational Digital Circuits," *IEEE Trans. on Computers*, Vol. C-21, No. 8, pp. 858-866, August, 1972.

[Shen *et al.* 1985] J. P. Shen, W. Maly, and F. J. Ferguson, "Inductive Fault Analysis of MOS Integrated Circuits," *IEEE Design & Test of Computers*, Vol. 2, No. 6, pp. 13-26, December, 1985.

[Smith 1978] J. E. Smith, "On the Existence of Combinational Logic Circuits Exhibiting Multiple Redundancy," *IEEE Trans. on Computers*, Vol. C-27, No. 12, pp. 1221-1225, December, 1978.

[Smith 1979] J. E. Smith, "On Necessary and Sufficient Conditions for Multiple Fault Undetectability," *IEEE Trans. on Computers*, Vol. C-28, No. 10, pp. 801-802, October, 1979.

[Timoc *et al.* 1983] C. Timoc, M. Buehler, T. Griswold, C. Pina, F. Scott, and L. Hess, "Logical Models of Physical Failures," *Proc. Intn'l Test Conf.*, pp. 546-553, October, 1983.

[Wadsack 1978] R. L. Wadsack, "Fault Modeling and Logic Simulation of CMOS and MOS Integrated Circuits," *Bell System Technical Journal*, Vol. 57, pp. 1449-1474, May-June, 1978.

PROBLEMS

4.1 Find a circuit that has an undetectable stuck fault.

4.2 Is it possible to have a combinational circuit C with some signal S and test t such that t detects both S s-a-1 and S s-a-0? Give an example or prove it is impossible.

4.3 Determine the output function of the circuit of Figure 4.27 for the following faults:

a. AND bridge between inputs of gate G_1

b. The multiple fault $\{x_3$ s-a-1, x_2 s-a-0$\}$

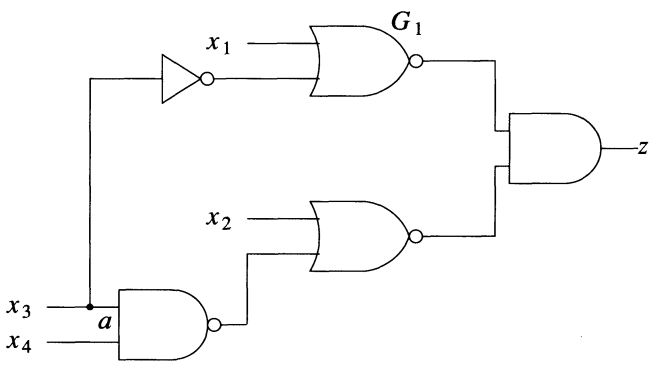


Figure 4.27

4.4 In the circuit of Figure 4.27 which if any of the following tests detect the fault x_1 s-a-0?

a. (0,1,1,1)

b. (1,1,1,1)

c. (1,1,0,1)

d. (1,0,1,0)

4.5 For the circuit of Figure 4.27 find a Boolean expression for the set of all tests that detect the fault:

a. x_3 s-a-0

b. x_2 *s-a-0*

c. x_2 *s-a-1*

4.6 For the circuit of Figure 4.28

a. Find the set of all tests that detect the fault *c s-a-1*.

b. Find the set of all tests that detect the fault *a s-a-0*.

c. Find the set of all tests that detect the multiple fault {*c s-a-1*, *a s-a-0*}.

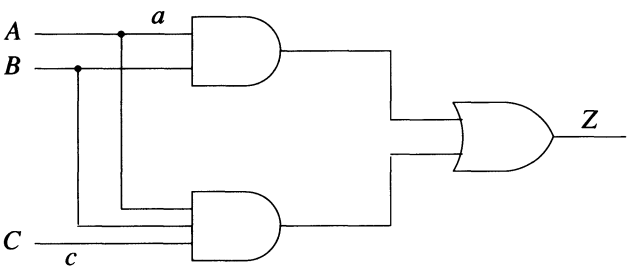


Figure 4.28

4.7 For the circuit of Figure 4.29

a. Find the set of all tests that detect the fault *a s-a-0*.

b. Find the set of all tests that detect the fault *b s-a-0*.

c. Find the set of all tests that detect the multiple fault {*a s-a-0*, *b s-a-0*}.

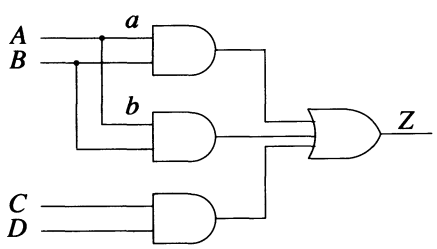


Figure 4.29

4.8 For the circuit of Figure 4.30

a. Find the set of all tests that detect the fault *b s-a-1*.

b. Find the set of all tests that distinguish the faults a s-a-0 and c s-a-0.

c. Find the set of all tests that distinguish the multiple faults $\{a$ s-a-0, b s-a-1$\}$ and $\{c$ s-a-0, b s-a-1$\}$.

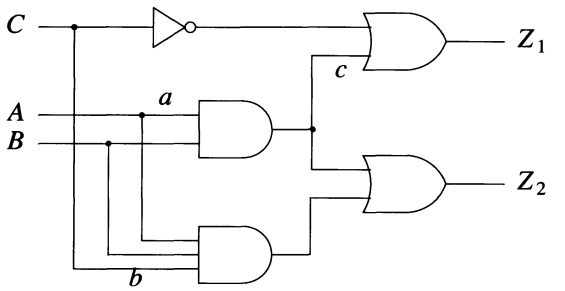


Figure 4.30

4.9

a. Find an example of a combinational circuit with two faults that are functionally equivalent under a test set but not functionally equivalent.

b. Prove that in a combinational circuit two faults functionally equivalent under a complete location test set are functionally equivalent.

4.10

a. Find a counterexample to the following statement:

"In a combinational circuit two faults f and g are functionally equivalent iff they are always detected by the same tests."

b. Find a class of combinational circuits for which the above statement is true and prove it.

4.11 Prove that in a combinational circuit, if two faults dominate each other, then they are functionally equivalent.

4.12 Prove that for combinational circuits fault dominance is a transitive relation; i.e., if f dominates g and g dominates h, then f dominates h.

4.13 Prove that in a fanout-free circuit, any pair of functionally equivalent faults are also structurally equivalent.

4.14 Analyze the relation between the detection of a stem line s-a-v and the detection of (single) s-a-v faults on its fanout branches for the case of nonreconvergent fanout.

4.15

a. For the circuit and the test in Figure 4.22, show that j s-a-0 is detected, but k s-a-0 and m s-a-0 are not.

b. For the circuit and the test in Figure 4.23, show that both x_1 s-a-0 and x_2 s-a-0 are detected, but x s-a-0 is not.

4.16 Let N be a combinational circuit composed only of NAND gates. Assume that every primary input has only one fanout. Show that a test set that detects all s-a-1 faults in N detects all s-a-0 faults as well.

4.17 Consider the circuit of Figure 4.31. Let f be the fault b s-a-0 and g be a s-a-1.

a. Does f mask g under the test 0110? Does f mask g under the test 0111?

b. Are the faults f and $\{f,g\}$ distinguishable?

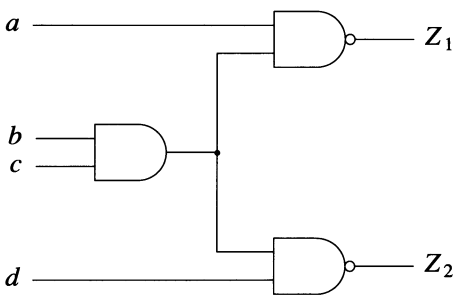


Figure 4.31

4.18 Prove that in an irredundant two-level combinational circuit, any complete test set for SSFs also detects all MSFs.

4.19 Let $Z(x)$ be the function of a single-output irredundant circuit N. Show that none of the SSFs in N can change its function from $Z(x)$ to $\overline{Z}(x)$.