UNIVERSITY OF WATERLOO
COMPUTER SCIENCE DEPARTMENT

Testability of CMOS Cells

J.A. Brzozowski

CS-85-31

September, 1985

TESTABILITY OF CMOS CELLS*

J.A. Brzozowski

Department of Computer Science
University of Waterloo
Waterloo, Ontario
Canada

ABSTRACT

The following types of faults are considered in CMOS cells: input and node stuck-at-0 and stuck-at-1 faults and transistor stuck-on and stuck-open faults. A fault is called clean if there is a sequence of input vectors resulting in one logic value in the good network and the complementary value in the faulty network, without the presence of any connections between V_{DD} and ground. A clean fault is called static if it can be detected by a single input vector and dynamic if the dynamic memory of an isolated output node must be used in order to detect it. We show that input stuck faults are static, transistor stuck-open faults are dynamic, and no other faults are clean. We then modify the CMOS cell by adding a control transistor and a control output. We prove that all of the faults studied are clean in the modified cell, including faults in the control transistor. This result holds for fully complementary CMOS cells.

1. Introduction

Considerable attention has been given to the problem of testing CMOS circuits during the past several years; see, for example, [B-C,C-V,E,H,M,M-B,W]. In one of the earliest papers on this subject, Wadsack [W] considered not only the "classical" input and output stuck-at-0 and stuck-at-1 faults, but also transistor stuck-open faults. The stuck open faults produce an isolated output node which retains its previous value because of the presence of capacitance. Thus the node has dynamic memory and the fault converts a combinational circuit into a

* This research was supported by the Natural Sciences and Engineering Research Council of Canada under grants No. A-1617 and A-0871.

sequential one. Wadsack mentioned, but did not pursue, transistor stuck-on faults which result in a connection between V_{DD} and ground.

A fault which can be detected by a single input vector, resulting in one logic value in the "good" cell and the complementary logic value in the faulty cell, will be called statically detectable or simply *static*. Wadsack showed that transistor stuck-open faults are not static, but can be detected by a sequence of two vectors, using the dynamic memory of an isolated output node; we will refer to such faults as dynamically detectable or simply *dynamic*.

Transistor stuck-open faults received further attention in [C-V,E,M,M-B]. McCluskey and Bozorgui-Nesbat [M-B] proposed the addition of a test transistor, a test line, and a charge/discharge line to each CMOS cell in order to facilitate testing stuck-open faults. Baschiera and Courtois [B-C] studied the detection of transistor stuck-on faults; their approach requires voltage measurements. Such an approach is less desirable than purely logical tests because one has to know the values of the various resistances in the network, and a fault that is detectable in one environment may become undetectable in another, depending on the switching threshold of the gate driven by the faulty gate [B-C]. Bryant and Schuster consider node stuck-at-0 and stuck-at-1 faults, in addition to transistor faults [B-S].

In this paper we study the detection of faults in CMOS cells. The following classes of faults are considered:

input stuck

output stuck

transistor stuck-open and stuck-on

node stuck

where stuck is either stuck-at-0 or stuck-at-1. We restrict our attention to *logic*

faults, where by a logic fault we mean one for which there exists an input sequence that results in one logic value in the good network and the complement of that value in the faulty network. Thus we do not consider performing any electrical measurements in order to detect a fault.

Part of this work appears in [B-SA].

We first introduce our notation and terminology by means of an example. Figure 1 shows the circuit diagram of a (fully complementary) CMOS logic gate or *cell*. A transistor labeled p is a p-type transistor, or simply *p-transistor*, which is conducting (ON) when its gate input is 0 and nonconducting (OPEN) when the gate input is 1. Dually, an *n-transistor*, labeled n, is ON when its gate input is 1 and OPEN when it is 0. Thus a p-transistor with gate input x may be considered as a normally closed contact, closed when $x' = 1$. Similarly, an n-transistor controlled by x corresponds to a normally open contact, closed when $x = 1$.

The top three transistors constitute the *p-part* of the cell C_1, and the bottom three —*the n-part*. Both parts may be viewed as two-terminal contact networks. The p-part has terminals 1 and f, and the variable t_1 represents the *transmission function* of the p-part with respect to these two terminals, i.e. $t_1 = 1$ iff there is a closed path from 1 to f. Similarly t_0 is the transmission function of the n-part with respect to terminals 0 and f. The contact network corresponding to cell C_1 of Figure 1 is shown in Figure 2.

In this paper we deal exclusively with fully complementary CMOS cells where $t_0 = t_1'$, i.e. the output node f is connected to 1 iff it is not connected to 0. In our example

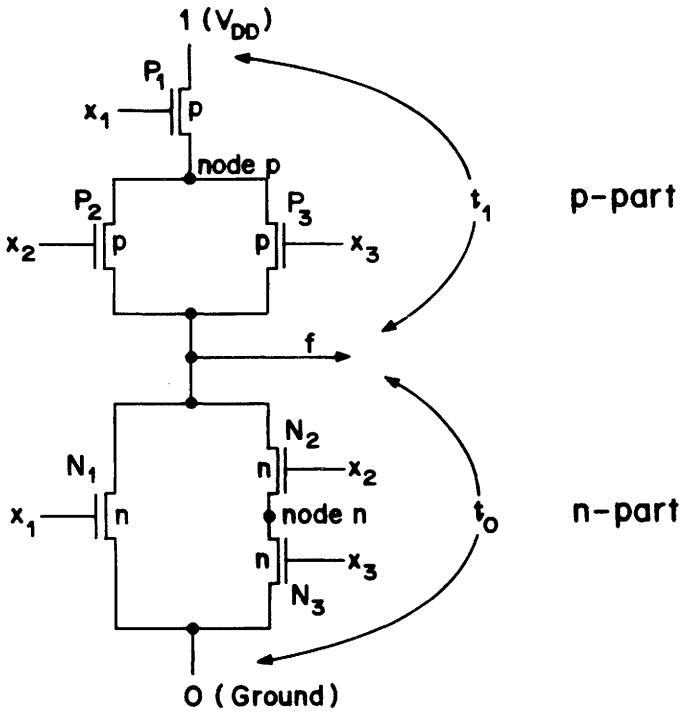

Figure 1. A CMOS cell C_1.

$$f = t_1 = x_1'(x_2' + x_3') = t_0' = (x_1 + x_2x_3)'.$$

For convenience, each p-transistor (n-transistor) with gate input x_i will be referred to as transistor P_i (N_i), for $i = 1, 2, 3$. Also, we refer to *input nodes* 0 and 1, *output node* f, and *internal nodes* p and n.

We now describe our interpretation of the various fault types with the aid of Figure 1. If x_i is stuck-at-0, then transistor P_i is permanently ON and transistor N_i is permanently OPEN. If x_i is stuck-at-1, P_i is OPEN and N_i is ON. If a node is stuck-at-0 (stuck-at-1) we may consider it to be an input node connected to ground (V_{DD}), i.e. another node labeled 0 (1). Note that the output f

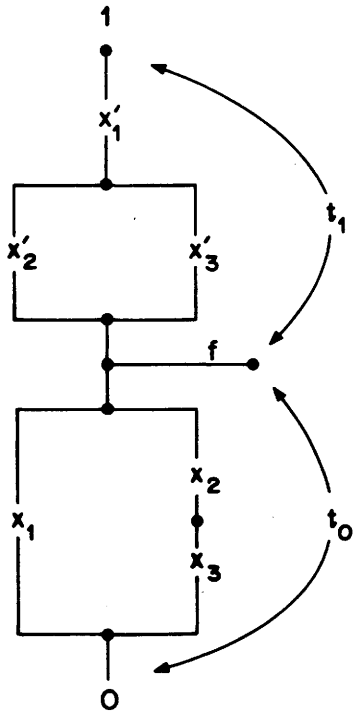

Figure 2. Contact network corresponding to C_1.

is also a node; the distinction between the output node and the internal nodes is made for technical convenience.

Throughout the paper we assume that only one fault may exist at any time.

In a "good" CMOS cell the state of the output node is always determined as follows. If there is a path from f to 0 then it necessarily consists of n-transistors, and the complementarity assumption guarantees that there is no path from f to 1. Thus the output terminal is "well-connected" to ground, and $f = 0$. Similarly, if there is a path from f to V_{DD} then it consists solely of p-transistors, and there is no path from f to ground. During certain fault conditions we will depart to some extent from this ideal behavior of a CMOS cell.

Specifically, we will assume that if a node is connected to 0 by a path consisting of p-transistors (rather than n-transistors) or of p-transistors and n-transistors, and it is not connected to 1, then this connection is sufficiently good to drive the node to 0. These assumptions are consistent with commonly used switch-level models such as that of Bryant [B].

The detection of faults is difficult in CMOS circuits because frequently it is necessary to create a path from 0 to 1 in the faulty circuit. For instance, if transistor N_1 is stuck-on there is a path from f to 0 in the faulty cell. To detect this fault we must have no path from f to 0 in the good cell. Thus we must set $x_1 = 0$ and either $x_2 = 0$ or $x_3 = 0$ or both. But any such condition creates a path from f to 1. Therefore, in the faulty cell there will be a path from 1 to 0, and the value of the voltage at f will depend on the relative sizes of the resistances from f to 1 and from f to 0. In general, the output voltage will then take on an intermediate value between V_{DD} and ground, and electrical measurements are required to detect the fault [B-C]. This may imply that the fault is not detectable by logical means alone.

In this paper any connection between 0 and 1 will be called a *fight*. We will call a fault *clean* if there exists a sequence of input vectors resulting in one logic value in the output of the good cell and the complementary value in the faulty cell, and there is no fight in the faulty cell at the time the last input vector is applied. Thus when a fault is clean, it is detectable by logic means alone; all other faults are called *unclean*. Clean faults can be either static or dynamic as explained earlier. The reader should note that some faults considered detectable by other authors are unclean here. For example, an output stuck-at-0 fault cannot be detected without a fight. We first show that very few faults are clean in standard CMOS cells. Then we introduce a modified cell, with one additional transistor and output, and prove that all the faults considered here are clean,

under the single-fault assumption.

The paper is structured as follows. Section 2 illustrates all the fault types using the example of Figure 1. Section 3 illustrates how faults can be made clean by introducing an additional transistor in the cell of Figure 1. Thus Sections 2 and 3 illustrate the main ideas of the paper, but only for one example. The remaining sections deal with general cells. In Section 4 we establish some mathematical background and describe certain assumptions that we make about the cells; basically we assume that there are no redundant inputs or transistors. In Section 5 we prove Theorem 1 which characterizes the faults in standard cells. Finally, in Section 6, we prove Theorem 2 which shows that all the faults considered in this paper become clean in the modified cell. This includes the faults associated with the added transistor.

2. Faults in a Standard Cell

We will use the cell of Figure 1 to introduce some additional terminology and illustrate various types of faults. We will denote by f, t_0, and t_1 the functions in the good cell C_1, and by f^*, t_0^* and t_1^* the corresponding functions in the faulty cell C_1^*. We consider the input to be a vector $x = x_1, x_2, x_3$ and will write $x = 011$ for $x_1 = 0, x_2 = 1, x_3 = 1$, etc.

Suppose input x_1 is stuck-at-0. When $x = 100$ we find that $f = 0$, $t_1^* = 1$, $t_0^* = 0$ and $f^* = 1$. Thus the fault is detected by this input vector, and it is clean since there is no fight in C_1^*. As mentioned before, we will say that such a fault is static. We will summarize this by the statement:

x_1 stuck-at-0 is static: $x = 100 \quad f = 0 \quad f^* = 1$

We will prove in Section 5 that all input stuck faults are static, and that no other fault is static.

Next consider the fault: transistor P_1 stuck-open. We show this fault is not static. First note that any input with $x_1 = 1$ would open P_1 in the good cell, and would fail to detect the fault. Therefore we must have $x_1 = 0$, if a test vector exists. The vector $x = 011$ gives $f = f^* = 0$, failing to detect the fault. The remaining possibilities are 001, 010 and 000. Any such vector results in $f = 1$, but $t_1^* = t_0^* = 0$ implying that f^* is isolated. If we first drive f^* to 0 by applying 100, f^* will remain 0 when it becomes isolated due to the dynamic memory of the node. We call this type of fault dynamic. It requires a sequence of two vectors. We will summarize these observations in the statement:

$$\begin{array}{lllll} P_1 \text{ stuck-open is dynamic:} & x = 100 & f = 0 & f^* = 0 \\ & x = 000 & f = 1 & f^* = \mathbf{0} \end{array}$$

where **0** (**1**) denotes an isolated node with value 0 (1).

We will prove in Section 5 that all transistor stuck-open faults are dynamic, and that there are no other dynamic faults. In fact, except for input faults and transistor stuck-open faults, there are no other clean faults. We give some examples of unclean faults below.

Suppose the output node is stuck-at-0. We must drive the good cell to $f = 1$ in order to detect this. But this implies that $t_1 = t_1^* = 1$, and there is a path from $f^* = 0$ to 1, i.e. a fight. Since both the p-part and the n-part are assumed to operate correctly, it is not possible to isolate the output node, i.e. dynamic tests also fail. Therefore this fault is unclean.

As a second example, consider P_2 stuck-on. To detect this fault we must use some input vector that separates node p from node f; otherwise the faulty network behaves like the good one. Thus we must use $x_2 = x_3 = 1$ which results in $t_1 = 0$ and $t_0 = 1$. Since the fault does not affect the n-part we also have $t_0^* = 1$. Now if $x_1 = 1$ we have $f = f^* = 0$, and, if $x_1 = 0$, we have a fight in C_1^* —the path $P_1P_2N_2N_3$. Here it is also impossible to isolate the output node. Consequently the fault is unclean.

As our last example in this section consider node p stuck-at-0. Suppose this fault is clean and that the final input vector is x. If $x_1 = 0$ in this vector there is a fight through transistor P_1. Thus we must have $x_1 = 1$, which implies that $t_1 = 0$ and $t_0 = 1$. Since the n-part in the faulty network is not affected by the fault, we also have $t_0^* = 1$, and there is a path from f^* to 0. The stuck-at-0 fault at p can only add another path to 0. Thus $f = f^* = 0$. Therefore the fault is unclean.

The reader can verify by similar arguments that the remaining faults are all unclean. The general case will be proved in Section 5.

3. Faults in a Modified Cell

In Figure 3 we show the cell of Figure 1, but with an additional control transistor P inserted in series between the p-part and the n-part. The gate input of P is the control input c. Note that there is an additional node g created; we assume that this node is observable, i.e. g is another output of the cell —the control output.

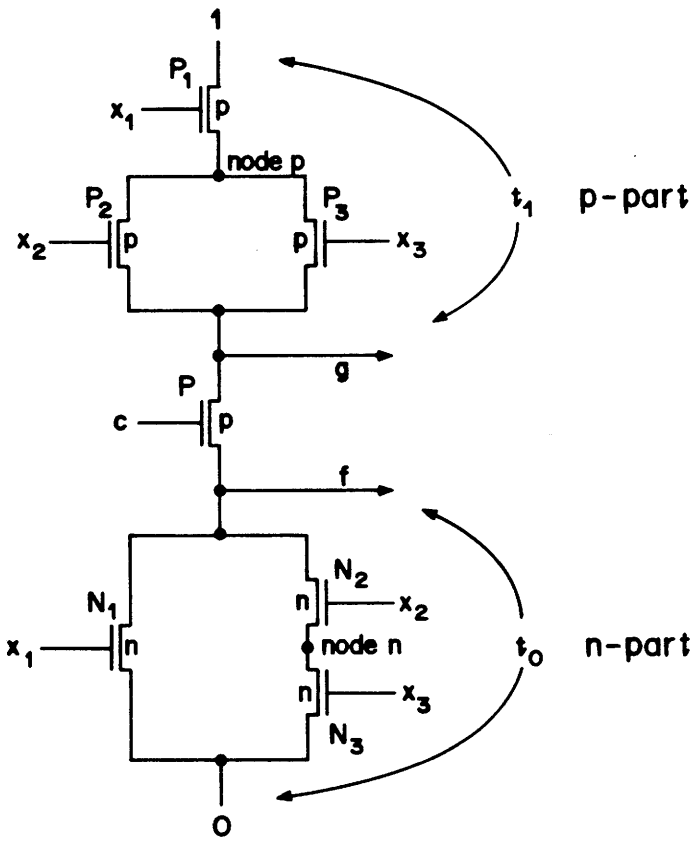

Figure 3. Modified cell $\tilde{C}_1$.

When $c = 0$, transistor P is ON and the modified cell $\tilde{C}_1$ behaves just like C_1. When $c = 1$ the two parts of $\tilde{C}_1$ are separated. It is this capability that results in clean faults in $\tilde{C}_1$. We consider several examples.

Suppose f^* is stuck at 0. One verifies that this fault is dynamic:

Output stuck-at-0 is dynamic: $c = 0$ $x = 000$ $f = 1$ f^*-fight

$$c = 1 \qquad x = 000 \qquad f = \mathbf{1} \qquad f^* = 0$$

Similarly, P_2 stuck-on becomes dynamic. Apply first $c = 0, x = 011$; then node g is connected to 0 through P, N_2, and N_3, and it is not connected to 1. Under our assumptions about the switch-level model, g is driven to 0. Next, $c = 1, x = 011$ isolates g, but g^* is connected to 1 through P_1 and P_2, and we have:

P_2 stuck-on is dynamic:	$c = 0$	$x = 011$	$g = 0$	g^*-fight
	$c = 1$	$x = 011$	$g = \mathbf{0}$	$g^* = 1$

Finally, consider node p stuck-at-0 in $\tilde{C}_1$. We find:

p stuck-at-0 is dynamic:	$c = 1$	$x = 000$	$g = 1$	g^*-fight
	$c = 1$	$x = 100$	$g = \mathbf{1}$	$g^* = 0$

where $g^* = 0$ results from the connection to 0 through P_2 or P_3.

These examples illustrate the key points. In Section 6 we will prove that all the faults of C_1 that we consider in this paper become clean in $\tilde{C}_1$. However, now that we have modified the cell, we must also consider faults associated with the control transistor. The reader will verify that:

P stuck-open is dynamic:	$c = 0$	$x = 111$	$f = 0$	$f^* = 0$
	$c = 0$	$x = 000$	$f = 1$	$f^* = \mathbf{0}$
P stuck-on is dynamic:	$c = 1$	$x = 111$	$f = 0$	$f^* = 0$
	$c = 1$	$x = 000$	$f = \mathbf{0}$	$f^* = 1$

g stuck-at-0 is dynamic:	$c = 1$	$x = 000$	$g = 1$	g^*-fight
	$c = 1$	$x = 111$	$g = \mathbf{1}$	$g^* = 0$
g stuck-at-1 is dynamic:	$c = 0$	$x = 111$	$g = 0$	g^*-fight
	$c = 1$	$x = 111$	$g = \mathbf{0}$	$g^* = 1$

Since c stuck-at-0 (stuck-at-1) is equivalent to P stuck-on (stuck-open), this exhausts all the control faults.

4. Mathematical Background

In this section we describe our mathematical model of CMOS cells and make certain assumptions about their behavior in order to eliminate degenerate and redundant designs. For our purposes, a *CMOS cell* is a switch network in the form of Figure 4. The external inputs, described by the vector $x = x_1, \ldots, x_n$ of input variables, are connected to the gate terminals of transistors in the two parts. The *p-part* is an arbitrary two-terminal network of p-transistors, and the *n-part* is any network of n-transistors that is *complementary* to the p-part. The transmission functions of these two networks are t_0 and t_1, and $t_0 = t_1'$ — as before.

The transistors will be labeled $P_1, \ldots, P_H$ in the p-part and $N_1, \ldots, N_K$ in the n-part. Similarly, the internal nodes will be labeled $p_1, \ldots, p_h$ in the p-part and $n_1, \ldots, n_k$ in the n-part; these do not include the input nodes 0 and 1 nor the output node f.

Let $B = \{0, 1\}$. Note that, for $a, b \in B$, $a < b$ iff $a = 0$ and $b = 1$. The partial order $\leq$ on B is extended in a natural way to B^n, the cartesian product of n copies of B:

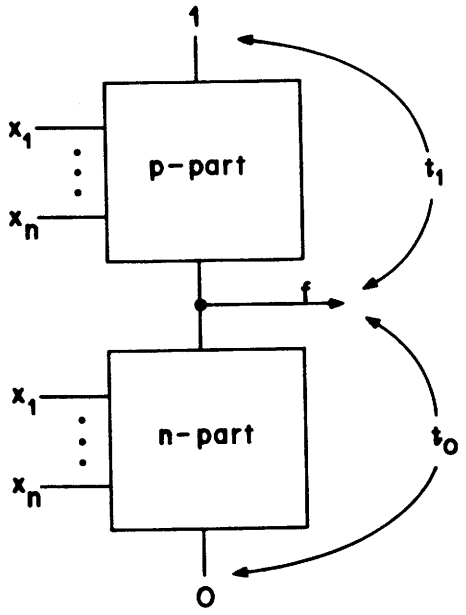

Figure 4. General CMOS cell.

$$a \leq b \text{ iff } a_i \leq b_i \text{ for } i = 1, \ldots, n,$$

where $a = a_1, \ldots, a_n$ and $b = b_1, \ldots, b_n$ are in B^n.

A boolean function $f : B^n \rightarrow B$ is *positive* iff there exists a sum of products (or product of sums) expression for f that does not contain any complemented variables. It can be verified [G] that f is positive iff, for all $a, b \in B^n$,

$$a \leq b \text{ implies } f(a) \leq f(b).$$

Similarly, f is *negative* if there is a sum of products expression for f that involves only complemented variables. A function f is negative iff

$$a \leq b \text{ implies } f(a) \geq f(b).$$

In any CMOS cell the transmission function t_1 is negative and t_0 is positive. Also $f = t_1 = t_0'$. Consequently the output f is always a negative function of the inputs.

We use the notation a_{i0} to denote some vector in B^n with the ith component equal to 0, i.e.

$$a_{i0} = a_1, \ldots, a_{i-1}, 0, a_{i+1}, \ldots, a_n.$$

Similarly,

$$a_{i1} = a_1, \ldots, a_{i-1}, 1, a_{i+1}, \ldots, a_n,$$

represents the same vector as a_{i0}, except that the ith component is 1.

In order to avoid degenerate cases of CMOS cells we make several (very weak) assumptions about them. The first condition simply states that there are no useless inputs.

Assumption 1. *In every CMOS cell the function f depends on each input. More formally, for each $i = 1, \ldots, n$ there exists $a_{i0} \in B^n$ such that $f(a_{i0}) \neq f(a_{i1})$.*

Assumption 1 is equivalent to the condition that, for each $i = 1, \ldots, n$, there exists $a_{i0} \in B^n$ such that

$$f(a_{i0}) = t_1(a_{i0}) = t_0'(a_{i0}) = 1,$$

$$f(a_{i1}) = t_1(a_{i1}) = t_0'(a_{i1}) = 0.$$

This is easily verified using the properties of negative and positive functions, since $a_{i0} < a_{i1}$, and $f(a_{i0}) < f(a_{i1})$.

Another consequence of Assumption 1 is the fact that f is not identically 0 or 1. For convenience, let $\bar{0}$ and $\bar{1}$ denote the vectors of length n whose components are all 0 and 1, respectively. Since f is negative, Assumption 1 implies that

$$f(\bar{0}) = 1 \text{ and } f(\bar{1}) = 0.$$

Let N be any two-terminal network of switches. The concept of tie-set [C] is useful in the analysis of such networks. A set T of switches is said to be a *tie-set* iff the network transmission function t is 1 when all the switches in T are closed, and $t = 0$ if any switch in T and all the switches not in T are open. Thus T is a minimal set, in the sense that no subset of T can cause t to be 1.

Given a tie-set T of p-transistors we define the *input of the tie-set* T to be the input vector a_T defined as follows. For each transistor P_j in T set its gate input to 0, and set all other inputs to 1. A tie-set T will be called *essential* iff there is only one path between f and 1 when $x = a_T$. We make a similar definition for tie-sets in the n-part.

Assumption 2. *The following applies to both the p-part and the n-part. Every transistor belongs to an essential tie-set.*

An example of a cell that violates Assumption 2 is one which has two transistors with the same gate input connected in parallel. Assumption 2 simply requires that each transistor is necessary, in the sense that it alone controls the closing and opening of the path between the output terminals of the appropriate part for at least one input vector.

Our final assumption requires that there be no useless nodes.

Assumption 3. *For every node p in the p-part there exists an input vector a_p such that f is connected to p but not connected to 1. Similarly, for every node n in the n-part there exists an input vector a_n that connects f to n but not f to 0.*

If Assumption 3 does not hold for a node p then, whenever f is connected to p, f is also connected to 1, i.e. p is also connected to 1. Hence node p could be identified with the input node 1 without changing the cell behavior. An example of a cell that does not satisfy Assumption 3 is one whose p-part consists of two transistors in series controlled by the same input. The internal node defined by the two transistors is then redundant, as is one of the transistors.

To show that these assumptions are not severe, we point out that for every negative function f that is not identically 0 or 1 there exists a CMOS cell satisfying Assumptions 1, 2, and 3. First find the sum of all prime implicants of f; one verifies that this is the unique irredundant sum of prime implicants of f, if f is positive or negative [B-Y]. Construct a parallel connection of series networks, where each series network corresponds to a prime implicant of f. This forms the p-part of the cell, and the n-part is constructed in a similar way. It is evident that this cell satisfies Assumptions 1, 2, and 3.

A CMOS cell satisfying Assumptions 1, 2, and 3 will be called *proper*.

5. Faults in Arbitrary Cells

We now generalize the observations of Section 2 to arbitrary CMOS cells.

Theorem 1. *Let C be an arbitrary CMOS cell. Then:*

(a) *A fault in input x_i is clean iff f depends on x_i. If such a fault is clean then it is static.*

(b) *A stuck-open fault in a transistor is clean iff the transistor belongs to some essential tie-set. If such a fault is clean it is not static but dynamic.*

(c) *All other faults are unclean.*

It follows that, in any cell satisfying Assumptions 1 and 2, the only clean faults are input faults and transistor stuck-open faults.

Proof:

(a) Suppose x_i is stuck-at-0. It is obvious that the fault cannot be detected (and hence is not clean) if f does not depend on x_i. If f does depend on x_i, there exists $a_{i1} \in B^n$ such that $f(a_{i1}) = 0$ and $f(a_{i0}) = 1$. Since the fault amounts to replacing a_{i1} by a_{i0}, we have $f^*(a_{i1}) = f(a_{i0}) = 1$. Hence the fault is static.

A similar argument holds if x_i is stuck-at-1.

(b) Suppose P_j with gate input x_i is stuck-open, and P_j belongs to an essential tie-set. Let a_{i0} be the tie-set input. We then verify:

P_j stuck-open is dynamic: $x = a_{i1}$ $f = 0$ $f^* = 0$

$x = a_{i0}$ $f = 1$ $f^* = \mathbf{0}$

Conversely, suppose the fault is clean. Then there must be an input a_{i0} such that $t_1(a_{i0}) = 1$ and $t_1^*(a_{i0}) = 0$. (But then $t_0(a_{i0}) = t_0^*(a_{i0}) = 0$ and f^* is isolated, proving that the fault is not static.) Now a_{i0} must establish a path from f to 1 through P_j. If any transistor in the path can be removed without breaking the path, remove it. Thus we eventually obtain a path from which no transistor can be removed, i.e. we have a tie-set containing P_j. Let the input of this tie-set be b_{i0}; it follows that $b_{i0} \geq a_{i0}$ and $t_1^*(b_{i0}) \leq t_1^*(a_{i0})$, since t_1 is negative. Thus $t_1^*(b_{i0}) = 0$ also, and P_j belongs to an essential tie-set.

A dual argument holds if N_j is stuck-open.

(c) We first prove that none of the remaining faults can be static. The case of output node stuck-at-0 has already been proved, by the general argument of Section 2, and the same argument holds for output stuck-at-1.

If P_j with gate input x_i is stuck-on, we must find an input a_{i1} such that $t_1(a_{i1}) = 0$ and $t_1^*(a_{i1}) = 1$. But this implies $t_0(a_{i1}) = t_0^*(a_{i1}) = 1$, and there is a fight. A similar argument holds if N_j is stuck-on.

If node p is stuck-at-0 and the test input a is such that $t_0(a) = 1$, the fault can only add another path to 0 and cannot be detected. Thus we must have $t_1(a) = 1$. If there is no path from f to p, the fault cannot be detected. However, if there is such a path, there is a fight.

Similar arguments apply to the three remaining types of node faults.

Finally, notice that output faults, node faults and transistor stuck-on faults cannot possibly lead to an isolated output node, and hence cannot be dynamic. □

6. Modified Arbitrary Cells

The results of Section 3 are now generalized to arbitrary CMOS cells. The general form of the modified cell is shown in Figure 5, where the p-part and the n-part are exactly the same as in Figure 4, and one p-type control transistor P with gate input c has been added in series with the two parts; this also adds a new node g. As before, t_0 and t_1 remain the transmission functions of the n-part and the p-part respectively. The cell output is f, but we now have an additional "control" output node g.

We point out that an n-transistor could be used equally well as the control transistor. The reader can easily verify that the same results hold if the appropriate modifications are made.

A modified cell is said to be *proper* iff the corresponding cell without the control transistor P is proper. We will refer to any standard CMOS cell as C and to its modified version as $\tilde{C}$.

Theorem 2. *All the faults considered in this paper are clean in any modified proper CMOS cell $\tilde{C}$. This includes faults associated with the control transistor P and the control node g.*

Proof: We have several cases:

Faults which are clean in C

When $c = 0$ transistor P is on, and the modified cell $\tilde{C}$ has the same behavior as C, when we consider the inputs to be $x_1, \ldots, x_n$ and the output to be f. Since we are concerned only with single faults, the control transistor and

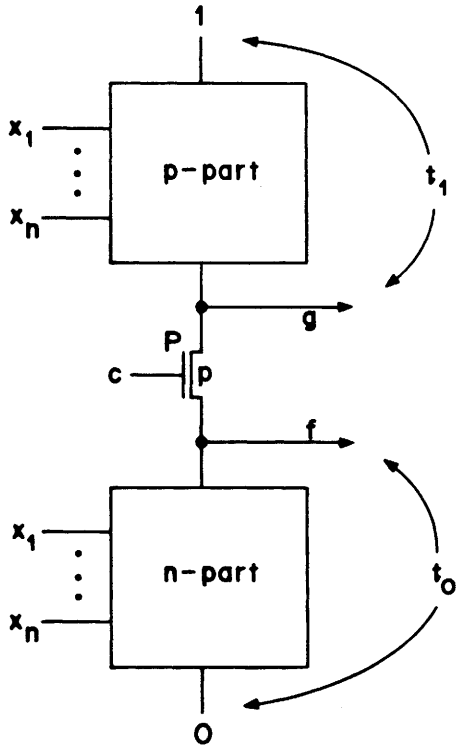

Figure 5. Modified cell $\tilde{C}$.

control node cannot have a fault if one exists elsewhere. Hence all the faults that are clean in Theorem 1 are detectable in the same way in $\tilde{C}$.

Output faults

The following is easily verified:

Output stuck-at-0 is dynamic:	$c = 0$	$x = \bar{0}$	$f = 1$	f^*-fight
	$c = 1$	$x = \bar{0}$	$f = \mathbf{1}$	$f^* = 0$

Output stuck-at-1 is dynamic:	$c = 1$	$x = \bar{1}$	$f = 0$	f^*-fight
	$c = 1$	$x = \bar{0}$	$f = \mathbf{0}$	$f^* = 1$

Transistor stuck-on faults

Suppose P_j controlled by x_i is stuck-on. By Assumption 2 P_j belongs to some essential tie-set T; let its input be a_{i0}. Then:

P_j stuck-on is dynamic:	$c = 0$	$x = a_{i1}$	$g = 0$	g^*-fight
	$c = 1$	$x = a_{i1}$	$g = \mathbf{0}$	$g^* = 1$
N_j stuck-on is dynamic:	$c = 0$	$x = b_{i0}$	$f = 1$	f^*-fight
	$c = 1$	$x = b_{i0}$	$f = \mathbf{1}$	$f^* = 0$

where b_{i1} is the input of an essential tie-set of N_j.

Node faults

Suppose node p is stuck-at-0. By Assumption 3 there exists an input vector a_p, such that f is connected to p, but p is not connected to 1. One verifies the following:

p stuck-at-0 is dynamic:	$c = 1$	$x = \bar{0}$	$g = 1$	g^*-fight
	$c = 1$	$x = a_p$	$g = \mathbf{1}$	$g^* = 0$
p stuck-at-1 is dynamic:	$c = 0$	$x = \bar{1}$	$g = 0$	$g^* = 0$
	$c = 1$	$x = a_p$	$g = \mathbf{0}$	$g^* = 1$

Node faults in the n-part are treated similarly.

Control faults

The arguments presented in Section 3 are completely general, if we replace 000 by $\bar{0}$ and 111 by $\bar{1}$; hence that proof applies also to arbitrary cells.

This concludes the proof. □

In summary, Theorem 1 states that very few faults are clean in a standard CMOS cell, and this makes testing difficult. Theorem 2 shows that all the faults are clean in the modified version of the cell. Thus the modified design is much better from the point of view of testability. The cost of using modified cells should not be prohibitive. Adding the extra transistor to a cell with several p-transistors and several n-transistors represents a relative small increase in area. Also, this approach can be easily combined with built-in self test techniques [M-B].

References

[B-C] Baschiera, D., and Courtois, B., "Testing CMOS: A Challenge," *VLSI Design, V*, No. 10, October 1984, 58-62.

[B] Bryant, R.E., "A Switch-Level Model and Simulator for MOS Digital Systems," *IEEE Trans. Computers, C-33*, No. 2, February 1984, 160-177.

[B-S] Bryant, R.E., and Schuster, M.D., "Performance Evaluation of FMOSSIM, a Concurrent Switch-Level Fault Simulator," *Proc. 22nd Design Automation Conference*, June 1985, 715-719.

[B-SA] Brzozowski, J.A., and Sayed, M., "Design of Testable CMOS Cells," *Proc. 1985 Canadian Conference on VLSI*, November 1985.

[B-Y] Brzozowski, J.A., and Yoeli, M., *Digital Networks*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1976.

[C] Caldwell, S.H., *Switching Circuits and Logical Design*, John Wiley & Sons, Inc., New York, 1958.

[C-V] Chiang, K.W., and Vranesic, Z.G., "On Fault Detection in CMOS Logic Networks," *Proc. 20th Design Automation Conference*, June 1983, 50-56.

[E] Elziq, Y.M., "Automatic Test Generation for Stuck-Open Faults in CMOS VLSI," *Proc. 18th Design Automation Conference*, June 1981, 347-354.

[G] Gilbert, E.N., "Lattice Theoretic Properties of Frontal Switching Functions," *J. of Mathematics and Physics, 33*, 1954, 57-67.

[H] Hayes, J.P., "Fault Modelling," *IEEE Design and Test of Computers*, April 1985, 88-95.

[M] Mandl, K.D., "CMOS VLSI Challenges to Test," *Proc. IEEE International Test Conference*, October 1984, 642-648.

[M-B] McCluskey, E.J., and Bozorgui-Nesbat, S., "Design for Autonomous Test," *IEEE Trans. on Computers, C-30*, No. 11, November 1981, 866-875.

[W] Wadsack, R.L., "Fault Modeling and Logic Simulation of CMOS and MOS Integrated Circuits," Bell System Tech. J., May-July 1978, 1449-1474.