THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 60, No. 7, September 1981
Printed in U.S.A.

Accurate Logic Simulation Models for TTL Totempole and MOS Gates and Tristate Devices

By Y. L. LEVENDEL, P. R. MENON, and C. E. MILLER

(Manuscript received December 16, 1980)

The two logic values, 0, 1, and the unknown, are not sufficient for accurately simulating the behavior of TTL totempole and MOS gates and tristate devices. Furthermore, the classical fault modes (output stuck and input open) are not sufficient to cover the faulty behavior of MOS devices. A previous solution to the simulation modeling required the addition of pseudo gates, which have no physical meaning. This paper develops methods of modeling fault-free and faulty tristate devices for logic simulation. The model does not require any additional circuitry, but the existence of a simulator capable of simulating any number of logic values is assumed.

I. INTRODUCTION

A component finding wide usage in the bus-oriented architecture of today's computer systems is the tristate driver. A typical arrangement is shown in Fig. 1. In this arrangement there are several drivers. Only one driver can be enabled at a time and "talk" to the bus. The receivers capture the information on the bus.

In transistor-transistor logic (TTL) technology, tristate devices allow bus wiring, previously obtained only with conventional open collector output TTL. They also allow the use of active pull up to charge the large capacitances associated with the bus. This feature, not available with conventional bus wiring technique, speeds up the operation of the bus. In MOS technology, similar effects can be achieved with lower power requirements. Here, we shall consider TTL totempole, CMOS, PMOS, and NMOS bistate and tristate devices and show the similarities and differences.

In tristate technology, several malfunctions due to the presence of faults or to a wrong utilization of the bus may occur. These malfunc-

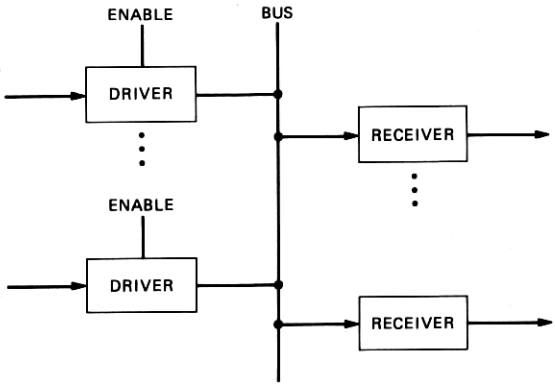

Fig. 1—A tristate bus system.

tions may invalidate test results or damage the components. Therefore, it is important to simulate accurately the operation of faulty and fault-free circuits containing buses, and other tristate devices.

It has been shown that the classical fault modes (output stuck, input open) are not sufficient to cover the faulty and fault-free behavior of CMOS devices.[1,2] One attempt has been made to map these fault effects into classical stuck-type faults by adding circuitry to the fault-free circuit. This additional circuitry is used to provide the faulty and fault-free circuits with memory properties, which exist in CMOS devices under certain conditions. This mapping allows the use of a fault simulator, which simulates only classical faults, for simulating faults in CMOS devices. In fact, the limitations of the available simulator was a constraint on the proposed modeling. Although modeling of the fault-free tristate devices also used similar added circuitry, the effect known as overlap or bus contention which may damage the devices, was not covered by this model.

This paper develops methods of modeling fault-free and faulty tristate devices for logic simulation, without additional circuitry. However, it assumes the existence of a simulator capable of simulating any number of logic values. Both the memory properties of MOS devices and the effects of bus contention are shown to be modeled accurately by the proposed method.

1.1 TTL tristate technology

Consider the tristate inverter of Fig. 2, in which A is the data input and E is the enable lead. The device is enabled and acts as an inverter, when $E = 0$ as shown in Table I. When the inverter is disabled, it assumes a high impedance, namely the impedance between V_o and V_{CC}, and the impedance between V_o and V_G is extremely large. V_o, V_{CC}, and V_G are the output, supply and ground voltages, respectively.

Table I—Truth table for tristate inverter

A	E	V_o
0	0	1
0	1	Disabled
1	0	0
1	1	Disabled

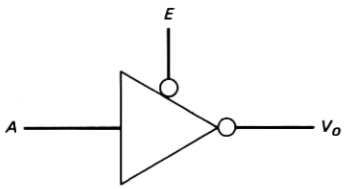

Fig. 2—Tristate inverter.

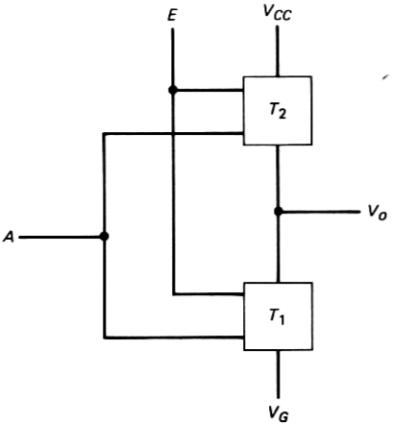

Fig. 3—TTL tristate inverter model.

The TTL tristate inverter can be modeled as the connection of two functions T_1 and T_2 (Fig. 3) which are controlled by lines A and E. T_1 and T_2 can be either conducting (on) or nonconducting (off) and they operate according to Table II. The device is in the high-impedance state when both T_1 and T_2 are off. In a tristate bus system, several tristate devices are wired together and the system operates safely if at most one device is enabled at one time (Fig. 1).

Two problems have emerged in tristate bus technology and they are associated with the structure of the tristate devices. The first difficulty concerns a disabled tristate device and its ability to source or sink current depending on the value of its output voltage. These two cases are illustrated in Fig. 4. One can identify a voltage V_{th}, such that, if $V_o > V_{th}$, then T_2 acts as a current source, and if $V_o < V_{th}$, T_1 acts as a sink. Threshold voltage V_{th} is a voltage between V_{CC} and V_G, which is determined by the output properties of the device.

Table II—Truth table for inverter model

A	E	T_1	T_2
0	0	Off	On
0	1	Off	Off
1	0	On	Off
1	1	Off	Off

If a receiver is present on a bus—i.e., it is connected to V_o—the driver will source or sink current and, as a result, the output V_o may reach the input threshold voltage V_{th} of the receiver. The output of a

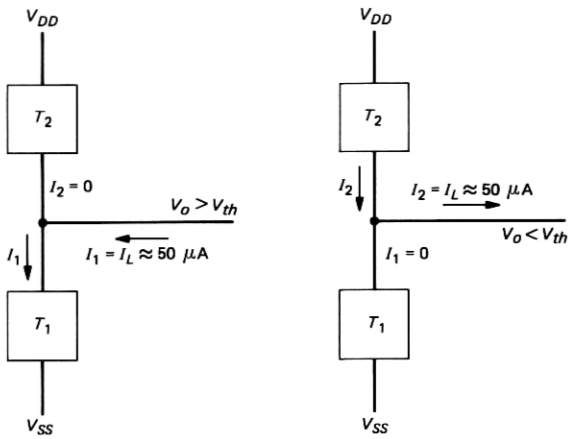

Fig. 4—Disabled bus. (a) Sink. (b) Source.

receiver with the input voltage equal to V_{th} is unknown and in fact the output may oscillate due to small variations around V_{th}. Normally, after all the driving devices become disabled, the existence of the leakage current I_L will destroy the previous logic value of a bus, and the bus will "float."

A second problem occurs when at least two tristate devices feeding a bus are simultaneously enabled and are in opposite active logic states (Fig. 5). Under this condition, the bus voltage may be anywhere in the range between the active logic levels and the currents may become extremely large (Fig. 5b). The actual value of V_o and I_o can be

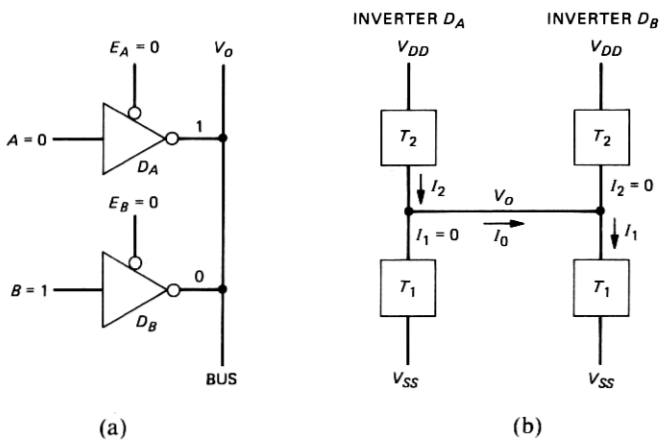

Fig. 5—Bus conflict.

determined from the current-voltage characteristics of the two devices. This condition, called overlap, may cause excessive device heating resulting in device failure or slowly degrade the device, causing a decrease in life.

In simulation, it is important to correctly model the effects of these two problems, so that a simulation user can be warned of the existence of potential difficulties. For instance, a primary input-output bus should be disabled (floating) before a test can be applied to it and enabled before the result of a test can be read from it. Also, an incorrect design or test sequence may cause overlaps on buses and the simulation should produce a warning.

II. TTL TOTEMPOLE, CMOS GATES, AND TRISTATE TRANSMISSION DEVICES

2.1 Pull-up and pull-down functions

Consider the CMOS and TTL implementations of an inverter (Fig. 6). They have a common structure, which can be generalized by the diagram of Fig. 7 for multiple input gates. This structure is composed of a pull-up function (PUF), a pull-down function (PDF), and an integrator (I). The PUF and PDF depend upon the input values $x_1, \cdots, x_n$ and the integrator produces the output Y depending upon Y_U and Y_D. Also the PUF and the PDF are complementary and cannot produce the

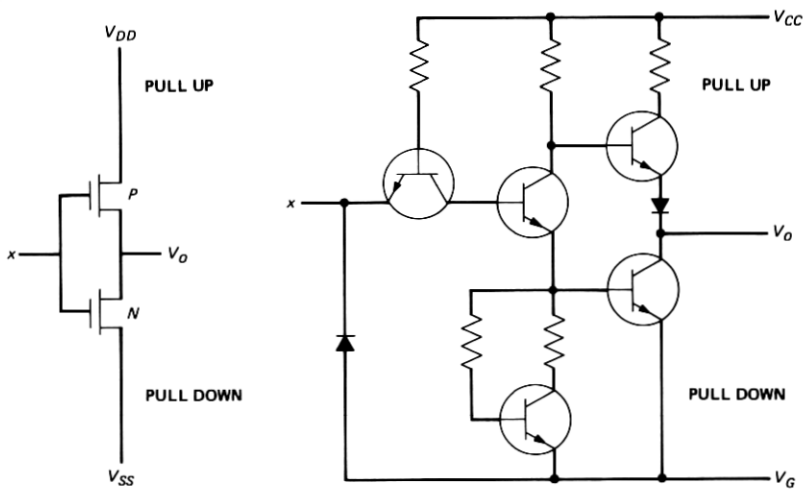

Fig. 6—(a) CMOS inverter. (b) TTL totem pole inverter.

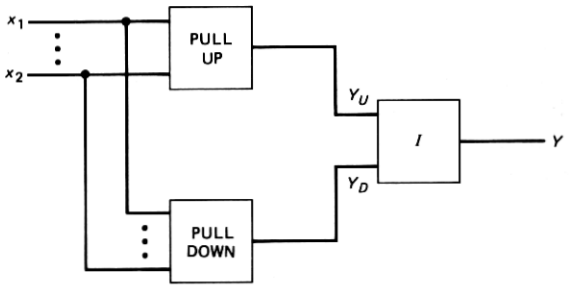

Fig. 7—General model for CMOS and TTL totem pole devices.

same logic value under normal circumstances. By convention, Y_U or Y_D has the value 1(0), when the PUF or the PDF is on(off). The integrator I has the behavior of Table III, except in the case of malfunctions. As an example, consider the CMOS, NAND gate of Fig. 8. The junctions P_1 and P_2 realize a NAND function, whereas N_1 and N_2 realize an AND function.

Table III—Truth table for integrator

Y_U	Y_D	Y
0	0	Impossible
0	1	0
1	0	1
1	1	Impossible

2.2 Tristate devices

A general CMOS or TTL tristate device can be modeled as in Fig. 9. The symbol E represents an enable line, and both PUF and PDF can be simultaneously disabled. Under normal conditions, the PUF and PDF cannot be simultaneously active. The integrator I is described in Table IV.

Table IV—Truth table for tristate integrator

Y_U	Y_D	Y
0	0	High impedance
0	1	Logic 0
1	0	Logic 1
1	1	Impossible

Usually, tristate devices are used in the mode shown in Fig. 10. In the illustration, $E_1, E_2, \cdots E_m$ are the enabling lines. There are several interesting cases, namely

(*i*) All the devices are disabled.

(*ii*) One device is enabled.

(*iii*) Two or more devices are enabled with opposite logic values.

These three cases lead to different impedance situations on the bus and they are represented in Table V. In the context of simulation, it is possible to find a fourth situation, when the impedances are not known. This can be caused by an unknown value on the enable line E of a device.

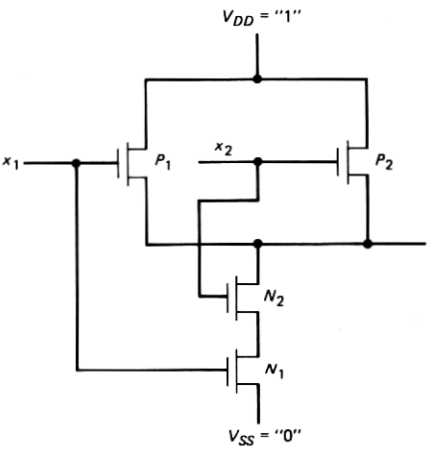

Fig. 8—CMOS NAND gate.

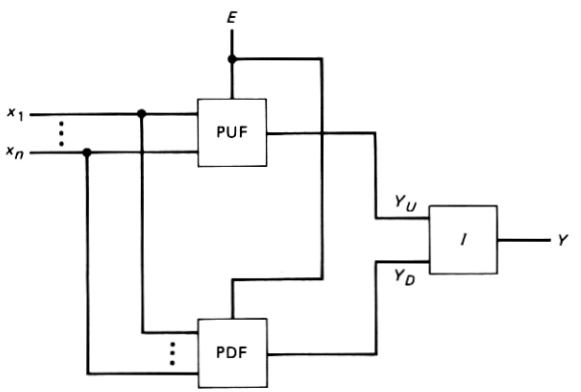

Fig. 9—General model for tristate devices.

Table V—Device states and bus impedances

Devices	Impedance
All disabled	High PUF, high PDF
One enabled	High PUF (PDE), low PDF (PUF)
Two enabled (conflict)	Low PUF, low PDF

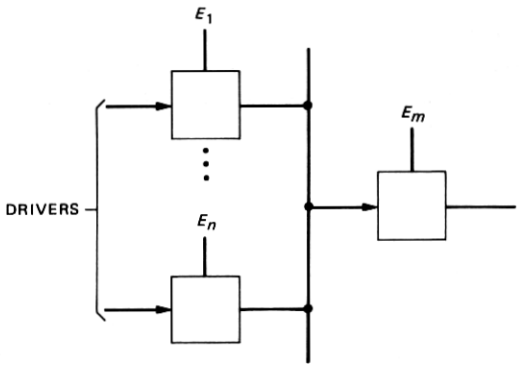

Fig. 10—Tristate devices connected to bus.

2.3 CMOS dynamic properties

The main difference between CMOS and TTL tristate devices is that the leakage currents in CMOS are extremely small compared to TTL leakage currents. Input currents for CMOS are also small. Therefore, if a CMOS device is first enabled and then disabled, the small capacitances on the buses will remain charged for a long period of time and the bus will appear to receivers as if it were remaining at the same logic value. Ultimately, the capacitance will be discharged, but if the rate of operation is sufficiently fast, the discharge time can be considered as infinite and the bus displays memory properties. However, if forced to an active logic value, the bus will immediately reach this value independent of this charged capacitance. In TTL tristate devices, the leakage currents being large, the discharge time becomes small and no memory is displayed.

2.4 Logic values and impedance

It should be clear from our preceding discussion that accurate simulation modeling of tristate devices requires two distinct concepts: logic value and impedance. Three logic values, 0, 1, and u, are widely used in simulation, the symbol u being used to represent unknown signal values.[3] Unknown signal values may be present because the initial values of some leads may be unknown or because of races or

oscillations. The effects of impedance on circuit behavior depends on the technology. For instance, a high impedance may appear as an unknown logic value in TTL tristate technology. On the other hand, an output which has a high impedance in CMOS technology will remember the logic value before the gate was disabled. Similarly, a conflict on a bus may appear as a 0, 1, or u depending on the technology.

During simulation of circuits containing tristate devices, it is important to be able to detect special situations like bus conflict. Tests that cause bus conflicts may result in damage to the devices and must be avoided. In the tester environment, the state of an output bus in the high-impedance state may be altered by the tester, invalidating the test. These considerations lead to the representation of the state of a line by a pair composed of the impedance value and the logic value. There will be four possible impedance values (Table VI). Therefore, we obtain 12 combinations of impedance and logic value (Table VII).

Table VI—Impedance values

PUF/PDF	Impedance	Impedance Representation
Both off	High	H
One on, one off	Regular	R
Both on	Conflict	C
One or both unknown	Unknown	U

Table VII—Combinations of impedance and logic values

	Pair	Description
1	R/0	Logic 0
2	R/1	Logic 1
3	R/u	Unknown with a low impedance
4	H/0	High impedance with previous state memory
5	H/1	High impedance with previous state memory
6	H/u	High impedance with unknown previous state memory
7	C/0	Conflict with logic 0 effect
8	C/1	Conflict with logic 1 effect
9	C/u	Conflict with logic u effect
10	U/0	Unknown impedance, logic 0 effect
11	U/1	Unknown impedance, logic 1 effect
12	U/u	Unknown impedance, logic u effect

These 12 logic combinations, which we shall call logic values in the context of simulation, represent a detailed analysis of tristate devices and any one of these corresponds to a possible situation. Two cases are illustrated in Fig. 11 and both cases display memory properties. In the first case (Fig. 11a), the enable line goes from 0 to u and the output goes from R/0 to U/0 (unknown impedance). In the second case (Fig. 11b), the enable line goes to 1 and the impedance goes from R to H, with the same logic value.

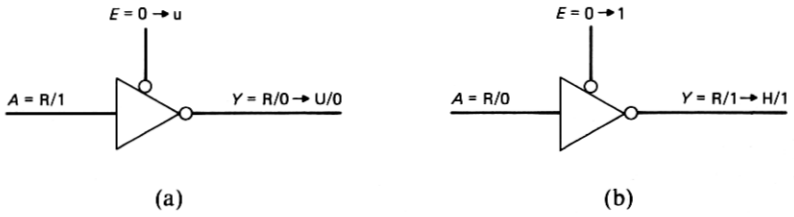

Fig. 11—Determination of impedance.

It is possible to reduce the number of values in Table VII at the expense of some information. The result is Table VIII, which shows two possible sets of logic values for TTL tristate devices. Z and a are synonyms for the pairs H/u and C/u, respectively. In set 2, pairs 3 and 12 are differentiated. Pair 12 is called a potential conflict (a*) and can occur in various situations (Fig. 12). In this case, the simulation will declare a potential bus overlap. In set 1, pairs 3 and 12 are not differentiated and some information may be lost. In TTL technology, the unused combinations correspond to impossible situations (e.g., pair 10) or to unpredictable situations (e.g., pair 8).

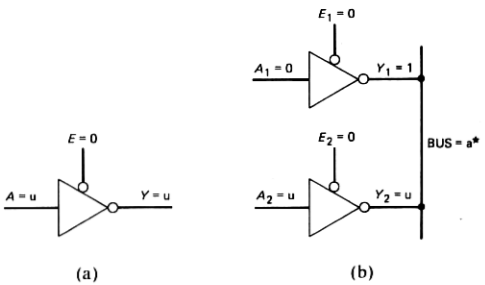

Fig. 12—Potential bus conflict.

Table VIII—Two sets of logic values for TTL tristate devices

Number	Pair	Set 1	Set 2
1	R/0	0	0
2	R/1	1	1
3	R/u	u	u
4	H/0	Unused	Unused
5	H/1	Unused	Unused
6	H/u	Z	Z
7	C/0	Unused	Unused
8	C/1	Unused	Unused
9	C/u	a	a
10	U/0	Unused	Unused
11	U/1	Unused	Unused
12	U/u	u	a*

In the case of CMOS technology, several additional values become meaningful (Table IX). The value Z0 (Z1) is used when a driver having the value 0(1) is disabled and remembers the previous logic value. The value u0(u1) is used when a driver was producing a 0(1) on a bus and the enable line becomes unknown. In all the three sets, pairs 7 and 8 could be used if the actual conflict voltage can be positioned as a 0 or a 1 when added structural knowledge is available.

Table IX—Logic values for CMOS devices

Pair	Set 3
1	0
2	1
3	u
4	Z0
5	Z1
6	Zu or Z
7	Unused
8	Unused
9	a
10	u0
11	u1
12	uu

The sets of values given in Tables VIII and IX can be used instead of impedance/logic value pairs. They are more economical in computer storage and more general; on the other hand, the pair representation may be more efficient, since the impedance is ignored in most of the gate evaluations (except the bus).

We shall illustrate the use of the pairs for a CMOS driver-inverter (Fig. 2). The behavior of the inverter is represented in Table X. A and E are the input and enable lines, respectively, and Y is the output of the device. The symbol x represents a "don't care" value.

Table X—Impedance-logic value table for CMOS-driver inverter

Previous Value of Y	AE = x0	AE = 01	AE = 11	AE = uu
0	H/0	R/1	R/0	U/u
1	H/1	R/1	R/0	U/u
u	H/u	R/1	R/0	U/u

A bus with any number of drivers may be calculated iteratively using Table XI. This table is symmetric with respect to the main diagonal. Its construction is illustrated by the case of three inverters producing $R/0$, $R/1$, and $H/1$, respectively, and wired to a bus. The

first pair produces C/u, and C/u is combined with H/1 to produce H/*, which can be approximated by H/u.

Table XI—Impedance-logic value table for tristate bus

	R/0	R/1	R/u	H/0	H/1	H/u	C/0	C/1	C/u	U/0	U/1	U/u
R/0	R/0	C/*	U/*	R/0	R/0	R/0	C/0	C/*	C/*	U/0	U/*	U/*
R/1	C/*	R/1	U/*	R/1	R/1	R/1	C/*	C/1	C/*	U/*	U/1	U/*
R/u	U/*	U/*	U/u	R/u	R/u	R/u	C/*	C/*	C/u	U/*	U/*	U/u
H/0	R/0	R/1	R/u	H/0	H/*	C/0	C/1	C/u	U/0	U/u	U/u	U/u
H/1	R/0	R/1	R/u	H/*	H/1	H/*	C/*	C/1	C/*	U/u	U/1	C/u
H/u	R/0	R/1	R/u	H/*	H/*	H/u	C/*	C/*	C/u	U/u	U/u	U/u
C/0	C/0	C/*	C/*	C/0	C/*	C/*	C/0	C/*	C/*	C/0	C/*	C/*
C/1	C/*	C/1	C/*	C/1	C/1	C/*	C/*	C/1	C/*	C/*	C/1	C/*
C/u	C/*	C/*	C/u	C/u	C/u	C/*	C/*	C/*	C/u	C/*	C/*	C/u
U/0	U/0	U/*	U/*	U/0	U/u	U/u	C/0	C/*	C/*	U/0	U/u	U/u
U/1	U/*	U/1	U/*	U/u	U/1	U/u	C/*	C/1	C/*	U/u	U/1	U/u
U/u	U/*	U/*	U/u	U/u	U/u	U/u	C/*	C/*	C/u	U/u	U/u	U/u

* Unknown (u) or technology-dependent value.

2.5 Refinement of unknown impedance values

Given that there are three basic impedance values, R, H, and C, the possibility of the enable signal being unknown during simulation introduces indeterminacy in the simulated impedance values. Seven impedance values can be used to represent the three known values and the four cases, where the impedance cannot be uniquely determined. The seven values are:

$$I_1 = H$$
$$I_2 = R$$
$$I_3 = C$$
$$I_4 = H \text{ or } R$$
$$I_5 = H \text{ or } C$$
$$I_6 = R \text{ or } C$$
$$I_7 = H \text{ or } R \text{ or } C$$

Figure 13 shows how the indeterminate simulated impedance may be generated. The impedance values are obtained by computing the impedances for a combination of the unknown signal values. For example, in Fig. 13c, it was possible to obtain I_5 because it was known that $E_1 = E_2 = 0$ or 1.

These seven impedance values preserve some information that would otherwise be lost. For instance I_5, I_6, and I_7 represent a potential overlap, whereas I_4 is definitely not an overlap. However, the overhead of dealing with a multiplicity of pairs may not be justified by the gain of information (21 impedance/logic value combinations).

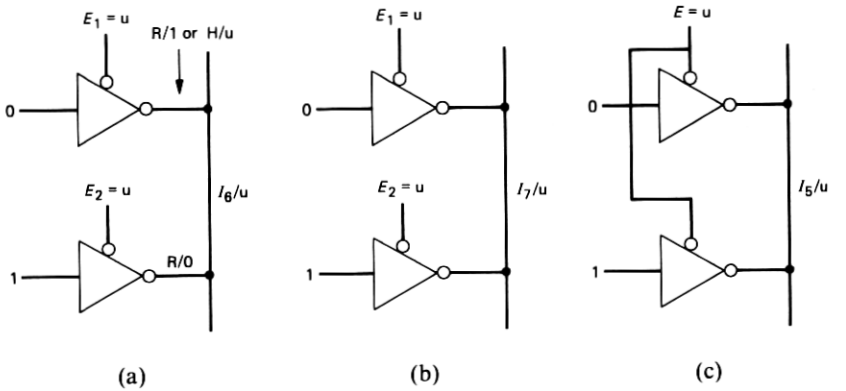

Fig. 13—Generation of indeterminate impedances.

In the above analysis, indeterminacy in the impedance and logic value are treated separately. This also results in some loss of information, which becomes apparent from the computed output of the upper tristate inverter in Fig. 13a. Although its output is known to be $R/1$ or H/u, it will be represented by I_4/u. Eliminating this problem would require creating one logic value for each subset of the set: $\{R/0,R/1,R/u,C/0,C/1,C/u,H/0,H/1,H/u\}$, excluding the empty subset. This system would have 2^9—1 or 511 logic values, which is impractical. In Fig. 14, the results would then become $\{C/u,R/0\}$ for case *a*, $\{C/u,H/u,R/1,R/0\}$ for case *b*, and $\{C/u,H/u\}$ for case *c*.

III. FAULT MODELS

Most of the faults that are peculiar to the devices considered in this paper can be simulated using the PUF-PDF model of Figs. 7 or 9.

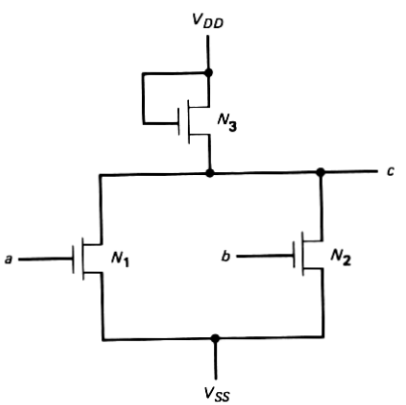

Fig. 14—NMOS NOR gate.

3.1 PUF and PDF faults and enable line faults

We shall consider first a special class of faults, where the PUF or PDF is enabled (disabled) when it is supposed to be disabled (enabled). This class is partitioned into four subclasses described in Table XII. Note that these are single faults.

Table XII—Class of PUF/PDF faults

Fault Subclass	Fault Effect on PUF	Fault Effect on PDF
I	Disabling	No effect
II	No effect	Disabling
III	Enabling	No effect
IV	No effect	Enabling

The fault effects on the output are described in Table XIII. The values marked with * are technology dependent and possibly unknown. Subclasses I and II cause a regular bistate gate to display tristate properties and a tristate device to be disabled when it is supposed to be enabled. Subclasses III and IV may cause a conflict (overlap) under the appropriate input values. In some sense, this fault class blurs the difference between a regular bistate gate and a tristate device. For this reason, we can use the same set of logic values for faulty and fault-free circuit modeling, namely a TTL set (set 1 or 2) or a CMOS set (set 3), for both tristate and bistate devices. The only difference between the two cases is that the high-impedance state will not be produced during the normal operation of a bistate device.

Table XIII—PUF, PDF, and output values

Fault Pull Up or Pull Down	Other Function	Output Value
$Y_U = 0$	$Y_D = 0$	$Y = H/*$
$Y_U = 0$	$Y_D = 1$	$Y = R/0$
$Y_U = 1$	$Y_D = 0$	$Y = R/1$
$Y_U = 1$	$Y_D = 1$	$Y = C/*$
$Y_D = 0$	$Y_U = 0$	$Y = H/*$
$Y_D = 0$	$Y_U = 1$	$Y = R/1$
$Y_D = 1$	$Y_U = 0$	$Y = R/0$
$Y_D = 1$	$Y_U = 1$	$Y = C/*$

Practically, the faults in each subclass can be obtained by shorted or open junctions in the PUF or PDF. We shall consider the example of Fig. 8 and summarize these subclasses of faults in Table XIV. In the context of concurrent fault simulation, the fault-injection mechanism

is extremely simple: a fault will be simulated if its effect on the gate is different from the fault-free circuit behavior, and this difference is measured over the set of possible logic values.

Another class of faults can appear in tristate devices and concerns the enable line. An enable line stuck at 0 (stuck at 1) will cause the device to be permanently enabled (disabled). A permanently disabled gate can be modeled by an output "stuck at Z." In the latter case, such a fault is characterized by a Z appearing at an output instead of a known logic value. The faults "enable line stuck at 1" and "output line stuck at Z" are equivalent.

Table XIV—Typical faults in a CMOS NAND gate

Fault Subclass	Fault Example	Input Values	Fault-Free	Fault
I	P_1 open	$x_1 = 0\ x_2 = 1$	1	Z*
II	N_2 open	$x_1 = 1\ x_2 = 1$	0	Z*
III	P_2 shorted	$x_1 = 1\ x_2 = 1$	0	a
IV	N_1 shorted	$x_1 = 0\ x_2 = 1$	1	a

* With memory of previous logic value.

3.2 PMOS and NMOS devices

The structure of an NMOS (PMOS) device is similar to the CMOS structure in that there is a pull-down function, but the pull-up function is a degenerate case of the CMOS pull-up function, namely it is permanently enabled and serves as a resistor (Fig. 14). The function $c = \overline{a + b}$ is implemented in Fig. 14.

We shall consider several fault modes, namely N_1 open, N_1 shorted, N_3 open, and N_3 shorted. The behavior of these four fault modes is represented in Table XV. The previous and present values of c are denoted by $c(-)$ and c, respectively. The behavior of faults N_1 open, N_1 shorted, and N_3 shorted is independent of $c(-)$. However, if N_3 is open, it is impossible to set c to a one, whereas the combination:

$$\begin{aligned} c(-) &= 0 \\ a &= 0 \\ b &= 0 \end{aligned}$$

produces a disabled output with new logic value equal to previous logic value ($c = Z0$). In spite of the behavioral differences, it is possible to model PMOS and NMOS devices using the same set of logic values as for CMOS devices.

3.3 Input-open faults in CMOS gates

It was shown earlier that a disabled CMOS tristate device displays certain memory properties that could be modeled using additional logic values $Z0$ and $Z1$. When an input to a CMOS gate is open, it is possible to produce these logic values in the faulty circuit. One method of modeling this is by setting the signal value at the site of the fault to a special value "propagating Zi" ($i = 0, 1, u$), and propagating the effect to the gate output. Denoting the propagating $Z0$ and $Z1$ by $PZ0$ and $PZ1$, respectively, we have the following conditions for the generation of these logic values at the site of the input-open fault: when the input changes from 1 or $Z1$ (0 or $Z0$) to 0(1), the faulty value of the input

Table XV—Typical faults in an NMOS NOR gate

			c			
a	b	$c(-)$	Open N_1	Short N_1	Open N_3	Short N_3
0	0	0	1	0	Z0	1
0	0	1	1	0	Impossible	1
0	1	0	0	0	0	a
0	1	1	0	0	0	a
1	0	0	1	0	0	a
1	0	1	1	0	0	a
1	1	0	0	0	0	a
1	1	1	0	0	0	a

becomes $PZ1$ ($PZ0$). With the introduction of these additional logic values for modeling the fault, we need a method of propagating them through gates. Table XVI shows the NAND function whose inputs are from the set {0, 1, $Z0$, $Z1$, $PZ0$, $PZ1$}. Since we are considering single faults, four entries in Table XVI are undefined.

This modeling may be applied to the NAND gate of Fig. 8. Consider the fault, junction N_1 open, and x_1 passing from 0 to 1 while $x_2 = 1$. The fault-free output will pass from 1 to 0 and the faulty output will remain at the value $Z1$, meaning that the fault may be detected after the change of x_1 to 1, if it is possible to register $Z1$ as the value 1.

Input-open faults in CMOS gates can also be treated as special types of faults that may produce $Z0$ or $Z1$ on gates outputs depending on gate type and present and previous input values. However, this would require treating input open faults on different types of gates differently. The proposed method presents a uniform way of inserting the effect of the input-open fault and only needs additional logic values during gate evaluation. These logic values introduced for modeling the fault do not themselves reach the gate output.

Table XVI—NAND function with propagating high-impedance states

	0	1	Z0	Z1	PZ0	PZ1
0	1	1	1	1	1	1
1	1	0	1	0	Z1	Z0
Z0	1	1	1	1	1	1
Z1	1	0	1	0	Z1	Z0
PZ0	1	Z1	1	Z1	—	—
PZ1	1	Z0	1	Z0	—	—

IV. CONCLUSIONS

A general model consisting of a pull-up function, a pull-down function and an integrator is proposed for modeling TTL totempole, CMOS, PMOS, and NMOS devices. It is shown that an accurate representation of the state of tristate devices and also certain bistate devices require not only the logic values but also impedance values. A set of 12 combinations of impedances and logic values is proposed, each of which can be represented by a single value or by an impedance/logic value pair. Speed-storage trade-offs will determine the choice of representation. The set of logic values needed is shown to be a technology-dependent subset of the 12 combinations represented. The proposed model covers all the known tristate fault-free and faulty effects and does not require any additional modeling gates.

REFERENCES

1. R. L. Wadsack, "Fault Modeling and Logic Simulation of CMOS and MOS Integrated Circuits," B.S.T.J., *57*, No. 5 (May-June 1978) pp. 1449–74.
2. R. H. Krambeck, private communication.
3. J. S. Jephson, R. P. McQuarrie, and R. E. Vogelsberg, "A Three-Value Computer Design Verification System," IBM System J., *8*, No. 3 (1969), pp. 178–188.
4. E. B. Eichelberger, "Hazard Detection in Combinational and Sequential Switching Circuits, IBM J. Res. Develop., *9*, No. 2 (March 1965), pp. 90–9.
5. E. G. Ulrich and T. Baker, "The Concurrent Simulation of Nearly Identical Digital Networks," Computer, *7*, No. 4 (April 1974), pp. 39–44.